Implementierung eines nichtlinearen Planungssystems

D I P L O M A R B E I T

von

cand. inform. Friedemann Kienzler

Universität Karlsruhe
Fakultät für Informatik
Institut für Prozeßrechentechnik und Robotik
Prof. Dr.-Ing. U. Rembold
Prof. Dr.-Ing. R. Dillmann

Juni 1990

Referent:	Prof. Dr.-Ing. U. Rembold
Korreferent:	Prof. Dr.-Ing. R. Dillmann
Betreuer:	Dipl.-Inform. A. Köhne

Ich erkläre hiermit eidesstattlich, daß ich die vorliegende Diplomarbeit selbständig und ohne unzulässige Hilfe angefertigt habe. Die verwendeten Quellen sind im Literaturverzeichnis vollständig angegeben.

Karlsruhe, den 30. Juni 1990.

- Friedemann Kienzler -

INHALTSVERZEICHNIS
==========

Kapitel 1: Einleitung

Ein Forschungsschwerpunkt auf dem Gebiet der Künstlichen Intelligenz (KI) bildet die Analyse des menschlichen Problemlöseverhaltens mit dem Ziel, selbiges durch technische bzw. "intelligente" Verfahren nachvollziehen zu können. Eine Problemstellung kann durch die Beschreibung einer Start- und einer zu erreichenden Zielsituation gegeben sein. Der Weg von Start- zu Zielsituation bildet die Lösung des Problems. Planen, eingeschränkt auf den Bereich der KI, bezeichnet nun die Tätigkeit, zur Lösung eines Problems eine Folge von Aktionen zusammenzustellen, die, wenn sie ausgeführt werden, das gegebene Problem lösen. Ein Plan, das Ergebnis eines Planungsprozesses, stellt eine Repräsentation von Handlungen dar, die einen möglichen Weg von der Start- zur Zielsituation beschreiben. Ein Plan wird entworfen, um die Lösung eines Problems zu steuern.

1.1 Aufgabenstellung bei dieser Diplomarbeit

Die Zielsetzung dieser Diplomarbeit besteht darin, ein nichtlineares Planungsprogramm gemäß der Beschreibung des bereits existierenden Planers TWEAK [Chapman 1987] zu implementieren, Schwachpunkte und Verbesserungsmöglichkeiten dabei aufzuzeigen und eine theoretische Analyse einer derartigen Vorgehensweise bei der Erstellung von Plänen durchzuführen.

1.2 Projekt-Umgebung

Diese Arbeit entstand im Rahmen eines aktuellen Projekts mit der Zielsetzung, verschiedene Planungstechniken (siehe 1.3) in der Planungs-Shell X3 zur Lösung von Planungs- und Konfigurierungsproblemen zu integrieren.

Das implementierte Planungssystem arbeitet zwar in der vorliegenden (Grund-) Version, also ohne Hinzunahme von Vorgehens- oder Strategiewissen in Form bereichsspezifischer Heuristiken, zu ineffizient, als daß komplexere Probleme damit gelöst werden könnten, aber es leistet insofern einen nützlichen Beitrag, daß es die prinzipielle praktische Verwendbarkeit des gewählten Planungsansatzes und der benutzten Formalismen bestätigt.

Es sei noch angemerkt, daß der Forschungstrend in Richtung solcher Planungs-Shells geht, die über verschiedene Methoden zur Plan-Erzeugung verfügen und durch Hinzugabe von speziellem Bereichswissen und Auswahl der anzuwendenden Planungsmethoden zu Planern in konkreten Domänen werden.

1.3 Verschiedene Planerstellungsmethoden in der KI

Nach einer nun folgenden einführenden Betrachtung bisher entwickelter Planungsmethoden, welche sich auf [Hertzberg 1989] stützt, sollen die Vorzüge und Nachteile der speziellen Technik des nichtlinearen Planens im nächsten Kapitel aufgezeigt werden.

Verschiedene Vorgehensweisen zur Erstellung eines Planes sind denkbar und auch bereits in konkreten Planungssystemen implementiert worden.

Vorneweg sei noch angemerkt, daß die verschiedenen Planungstechniken, die im Anschluß kurz erläutert werden, nicht in dieses grobe Schema einzuordnen sind, wie dies die untere Abbildung (Abb. 1.3-1) vielleicht zu suggerieren vermag. Vielmehr werden diese unterschiedlichen Ansätze zur Plan-Generierung - zumindest in den meisten mir bekannten Planungssystemen - in Mischformen eingesetzt. So kombiniert etwa Stefiks Planer MOLGEN [Stefik 1981a, Stefik 1981b] mehrstufiges Planen in der Ausprägung der Operatorabstraktion mit nichtlinearem Planen; als weiteres Beispiel läßt sich etwa das Planungssystem SIPE [Wilkins 1984, Wilkins 1986] anführen, das bei der Plan-Generierung grundsätzlich hierarchisch vorgeht, Planen als Inferenz aber auch als Strategie einsetzt, um in einer gegebenen Situation einen anzuwendenden Operator zu bestimmen.

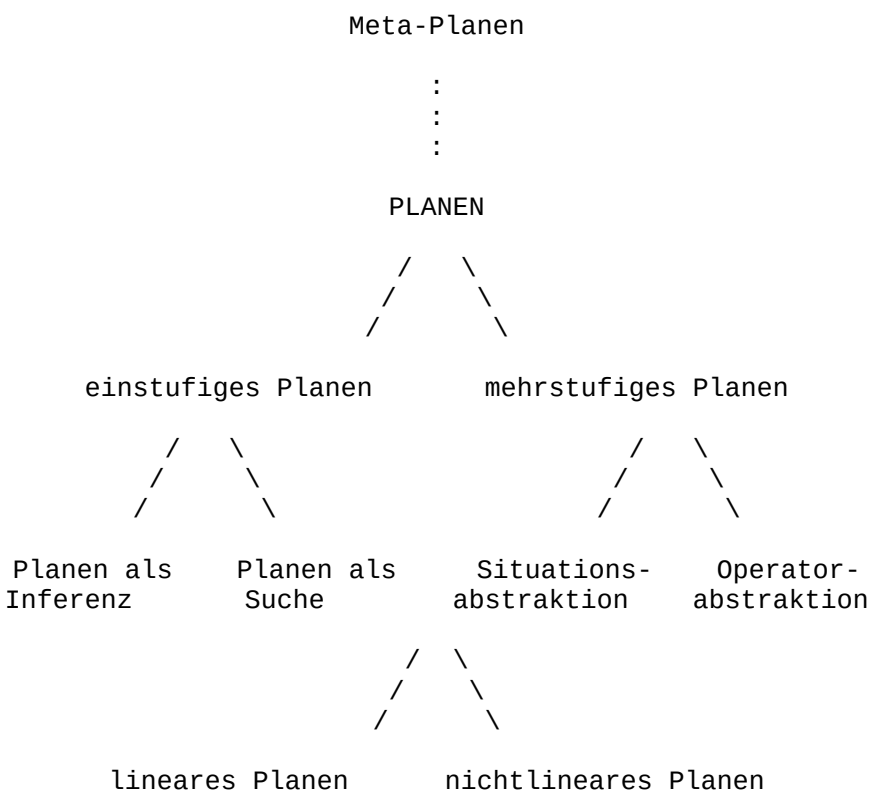


Abb. 1.3-1 Planerstellungsmethoden in der KI (Übersicht)

Grundsätzlich lassen sich zwei Methoden unterscheiden, nach denen ein Plan erstellt werden kann: Zum einen kann man von den verfügbaren Operatoren ausgehen und aus ihnen Folgen zu bilden versuchen, bis man schließlich eine gefunden hat, die das gestellte Problem löst; man spricht hierbei von einstufigem Planen. Zum andern kann man vom globalen Problem ausgehen, dieses erst im groben lösen und die Lösung dann schrittweise verfeinern, bis man bei den als elementar betrachteten und somit ausführbaren Operatoren angelangt ist; diese Vorgehensweise wird als mehrstufiges Planen bezeichnet.

Beim einstufigen Planen wird "bottom-up" vorgegangen: aus vorhandenen Elementaroperationen wird versucht, durch geeignete Kombinationen und Instantiierungen eine Operatorkonstellation zu gewinnen, die die gestellte Aufgabe löst. Der implementierte Planer sowie das als Vorlage dienende Planungssystem TWEAK generieren Pläne auf diese Weise. Der resultierende Plan wird gewissermaßen in einem Guß erzeugt, während sich die Planerstellung beim mehrstufigen (oder auch hierarchischen) Planen über mehrere Abstraktionsstufen erstreckt. Die Abstraktion kann sich hierbei auf Situationen oder Operatoren

beziehen. "Top-down" wird die Komplexität des Problembereichs "aufgebrochen". Die Planarbeit beschränkt sich zunächst auf das Wesentliche und greift erst nach und nach die Detailprobleme auf, ein anfängliches Plangerüst wird schrittweise in einen fertigen Plan umgesetzt. Planungssysteme wie MOLGEN und SIPE operieren gemäß diesem hierarchischen Ansatz.

Bei der Situationsabstraktion wird der Zustandsraum in verschiedenen Abstraktionsstufen repräsentiert. Zunächst wird ein Plan unter Berücksichtigung lediglich der wichtigsten Merkmale erzeugt und dieser dann durch Hinzunahme der weniger wichtigen Merkmale bis hin zur Ausführbarkeit auf primitiver Ebene verfeinert. Das Planungssystem TWAIN [Brooks 1985] setzt diese Planungstechnik, kombiniert mit der im folgenden beschriebenen, ein, indem verschiedenen (Roboter-)Aktionen unterschiedliche Prioritäten zugeordnet werden, so daß Interaktionen möglichst entgegengewirkt wird.

Bei der Operatorabstraktion wird ein Plan zuerst mittels abstrakter Operatoren erstellt. Das sind Operatoren, die nicht tatsächlich ausführbar vorhanden sind, sondern zu deren Ausführung in Wirklichkeit eine Folge konkreter Operatoren notwendig ist. Daraufhin findet dann eine Operator-Expansion statt, bis schließlich die Ebene der Elementaroperatoren erreicht ist. Planer, die u.a. diesen Planungsansatz wählen, sind z.B. MOLGEN und TWAIN, als "Klassiker" der Operator-Abstraktion gilt jedoch NOAH [Sacerdoti 1977].

Einstufiges Planen untergliedert sich weiter in Planen als Inferenz und Planen als Suche.

Bei der ersteren Methode, Planen als Inferenz, ist der Planerstellungsprozeß gleichzusetzen mit einer Beweisführung. Es geht darum zu beweisen, daß aus einer gegebenen Startsituation eine erwünschte Situation abgeleitet werden kann, die bestimmten Zielbedingungen genügt. Von Interesse ist nicht nur die Tatsache, daß ein derartiger Beweis erbracht werden kann, sondern auch und vor allem, wie die Beweisführung aussieht: Der Lösungsplan ergibt sich durch Zurückverfolgen des Beweises und Aneinanderfügen der verwendeten Operatoren.

Planen als Suche umgeht gewisse theoretische Schwierigkeiten, wie sie sich aus der Betrachtungsweise des Planens als reines Schließen in einem Kalkül unvermeidlich ergeben, dadurch, daß sich der Planungsprozeß als Suche innerhalb eines Zustandsraums, einer Menge von Situationen, welche sich jeweils durch Anwenden der verfügbaren Operatoren ergeben, darstellt. Es gilt, einen Weg von der Start- zu einer Zielsituation zu bestimmen, wobei Start- und Zielsituation durch die Problemstellung gegeben sind. Planen nach dieser Art ist der sukzessive Aufbau einer Folge atomarer Operatoren. Das für diese Sichtweise des Planens häufig zitierte Programm ist STRIPS [Fikes 1971].

Bei der Lösung eines Problems bietet sich generell die Vorgehensweise an, das Gesamtproblem in überschaubarere Teilprobleme zu untergliedern. Diese Teilprobleme sind dann einzeln zu lösen. Die resultierenden Lösungen können anschließend zusammengesetzt werden. Das zusätzliche Problem, das sich dabei stellt, besteht darin, daß Teilproblemlösungen interagieren, d.h. sich gegenseitig beeinflussen können.

Bei der linearen Planung müssen Teilpläne, zwischen denen eine Interaktion erkannt wurde, umgeordnet und dadurch zerstörte Teilziele durch zusätzliche Planarbeit wiederhergestellt werden. Das Planungsprogramm HACKER [Sussman 1975], um auch hierfür ein Beispiel anzuführen, versucht durch eben dieses Umordnen, auftretende Konflikte zwischen Plankomponenten zu beheben. Es wird jeweils eine bestimmte Operator-Sequenz mit festgelegter Reihenfolge der Einzeloperatoren ermittelt. Erweist sich eine Operator-Sequenz nicht zum Ziel führend, so wird eine andere Reihenfolge festgelegt.

Demgegenüber zeichnet sich nichtlineares Planen dadurch aus, daß die einzelnen Teilpläne bezüglich der zu erbringenden Ziele unabhängig voneinander betrachtet und Reihenfolgefestlegungen erst bei auftretenden Konflikten zwischen den

Teilplänen durchgeführt werden. Die einzelnen Teilpläne werden nicht linear aneinandergereiht, sondern zunächst nur partiell geordnet. Diese Ordnung wird im Laufe der Plangenerierung nach und nach erweitert und bei Bedarf verschärft, bis ein möglichst interaktionsfreier Gesamtplan entstanden ist. Als Planer, die diese Technik (u.a.) einsetzen, lassen sich hier MOLGEN und TWEAK anführen. Näheres zu diesem Planungsansatz findet sich im nächsten Kapitel.

Schließlich bietet sich noch die Möglichkeit an, über die eigentliche Problembereich-Planungsebene eine weitere Ebene zu spannen, auf der dann unabhängig von aller Planung im aktuellen Problembereich strategische Entscheidungen den Planungsprozeß an sich betreffend getroffen werden können. Hier wird entschieden, nach welchem abstrakten Muster in einer gegebenen Situation geplant bzw. weitergeplant werden soll. Man spricht von Meta-Planen. Das System, an dem Meta-Planen erstmals ausführlich untersucht wurde, ist MOLGEN.

1.4 Beispiele aus der Blockwelt

Zur Illustration werde ich immer wieder Beispiele aus der fiktiven Domäne Blockwelt anführen. Das bietet sich an, da diese der klassische Standardproblembereich der KI ist, in dem sich viele Probleme beim Planen in konkreten Domänen auf leicht verständliche Weise nachbilden lassen. Ein weiterer Grund besteht darin, daß der im Rahmen dieser Diplomarbeit implementierte Planer (siehe Kapitel 4) nicht den Anspruch erheben will, in realen Problembereichen zu operieren. Der Formalismus und Planungskern ist zwar mächtig genug, um von einem domänen-unabhängigen Planer sprechen zu können, jedoch sind die Repräsentationsmöglichkeiten stark eingeschränkt, so daß zu sehr von realen Objekten und Aktionen abstrahiert werden muß.

Ich gehe davon aus, daß der Problembereich der Blockwelt dem Leser bekannt ist.

Kapitel 2: Nichtlineares Planen (als Suche)

Dieses Kapitel soll zunächst die Charakteristika der Vorgehensweise beim nichtlinearen Planen (als Suche) grob aufzeigen. Im nachfolgenden Kapitel wird dann detailliert auf die Aktionsrepräsentation, den Planungsprozeß von der Problembeschreibung zum fertigen nichtlinearen Plan und die Planungsstrategie des Planungssystems TWEAK, der Vorlage des implementierten Planers, eingegangen.

2.1 Planen als Suche

Wie bereits im vorigen Kapitel erwähnt, ist dieses Planungsverfahren eine mögliche Ausprägung des einstufigen Planens.

Wonach wird beim Planen als Suche denn eigentlich gesucht?

Den Problembereich stellt man sich in einzelne Situationen untergliedert vor. Ausgehend von der Startsituation eines gegebenen Problems ergibt sich durch sukzessives Anwenden der verfügbaren und anwendbaren Operatoren eine Baumstruktur, deren Knoten Situationen entsprechen und deren Kanten als Anwendung von Operatoren zu interpretieren sind. Diese Darstellung des Zustandsraums erlaubt eine strukturierte Repräsentation der während der aktuellen Planung relevanten Information insofern, daß eine Partitionierung in Situationen erfolgt. Planen nach diesem Ansatz ist somit identisch mit der Suche in dem Zustandsraum der Situationen nach einem Weg von der Start- zu einer Zielsituation.

Jede Situation wird durch eine Reihe von Merkmalen, die in der betreffenden Situation gelten bzw. gelten sollen, beschrieben und stellt somit eine Ansammlung von Fakten dar. Eine mögliche Situationsbeschreibung für den Problembereich Blockwelt könnte etwa folgendermaßen lauten:

Situation S:
- Hand hält nichts,
- C steht auf Tisch,
- B steht auf C,
- B ist frei.

Durch Anwenden von Operatoren, den "Aktionsbausteinen" eines Planers, ändert sich der aktuelle Zustand der Welt, d.h. es findet ein Situationswechsel statt: Jede ausgeführte Handlung, sprich Operator-Anwendung, führt von einer Situation in eine andere. Dabei wird die Beschreibung der alten Situation in die neue hinüberkopiert, verändert nur um die Merkmale, die der angewandte Operator erzeugt bzw. löscht (Abb. 2.1-1). Sämtliche Änderungen, die durch die Anwendung eines Operators bewirkt werden, sind also durch die Nachbedingungen des jeweiligen Operators genau spezifiziert (STRIPS-Annahme; STRIPS assumption).

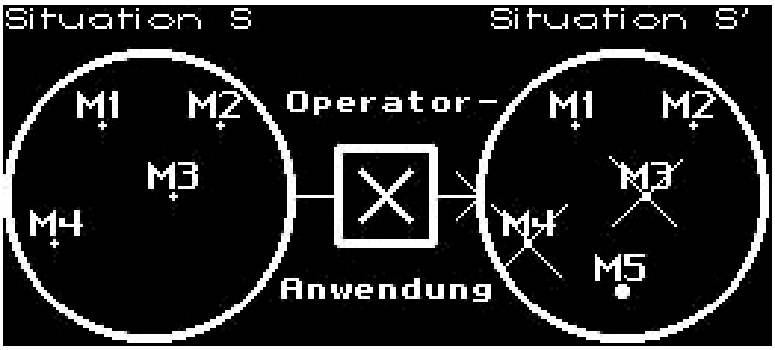


Abb. 2.1-1 Operator-Anwendung im Zustandsraum

Eine entscheidende Rolle spielt hierbei die Frage nach der Gültigkeit eines Merkmals in einer bestimmten Situation. Vor der Planausführung gelten nur die Merkmale, die bereits in der Ausgangs- oder Startsituation erfüllt sind. Nach Anwendung eines Operators ergibt sich folgendes Bild (die Angaben in den Klammern beziehen sich auf Abb. 2.1-1): Zunächst wird die Beschreibung der alten Situation (S), also der vor Anwendung des Operators (X), in die neue (S') hinüberkopiert (Merkmale M1, M2, M3 und M4). Die Situationsmodifikation (S -> S') findet dann dadurch statt, daß bestimmte Merkmale, spezifiziert in der sog. Weg-Liste (delete list) des Operators, nach dessen Ausführung nicht mehr gelten (Merkmale M3 und M4), dafür aber neue Merkmale, die in der Dazu-Liste (add list) aufgeführt sind, hinzukommen (Merkmal M5).

Die Planerstellung stellt sich somit als ein Prozeß dar, um nochmals auf obig gestellte Frage zurückzukommen, eine Situation zu bestimmen, in der die in der Problemstellung spezifizierten Ziele erfüllt sind, samt der Operatorfolge, um, ausgehend von der Startsituation, zu dieser Finalsituation zu gelangen. Der Zustandsraum, in dem gesucht wird, setzt sich also aus einer Menge von Situationen zusammen.

Mit diesem Ansatz bringt man das Rahmenproblem ganz gut in den Griff, das die Frage aufwirft, wie man in einem Planungsprogramm beschreibt, welche Merkmale des Problembereichs von den Operatoren nicht verändert werden. Beim Planen als Inferenz hingegen muß zusätzlich zur Beschreibung dessen, was durch die Anwendung eines Operators bewirkt wird, auch noch angegeben werden, wie der konstante "Rahmen" aussieht, vor dem die Änderungen erbracht werden, was also durch die Operator-Anwendung unverändert bleibt.

2.1.1 Repräsentation von Operatoren

Ein Operator (Abb. 2.1-2) wird spezifiziert durch eine Menge von Vorbedingungen, die erfüllt sein müssen, damit die Anwendung erfolgen kann, und eine Menge von Nachbedingungen, welche die Effekte definieren, also die Merkmale, die der Operator erzeugt (-> add list) bzw. löscht (-> delete list).

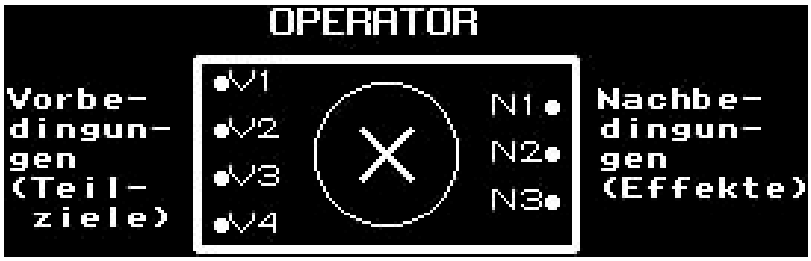


Abb. 2.1-2 Operator-Spezifikation

Wird ein solcher Operator zur Erfüllung eines Teilzieles in einen Plan eingebaut, so stellen seine Vorbedingungen neue Teilziele dar, die im weiteren Planungsverlauf zu erfüllen sind. Während der Plangenerierung werden also u.U. neue (Teil-)Ziele geschaffen, die zusätzlich zu den in der Zielsituation erwünschten Merkmalen durch entsprechende Aktionen erzeugt werden müssen. Daher könnte eine Strategie zur Erzeugung eines möglichst optimalen Lösungsplans für ein gestelltes Problem so aussehen, daß zunächst im bereits generierten Plan nach Aktionen zur Erfüllung irgendwelcher Teilziele gesucht wird, bevor neue Operatoren in den bestehenden Plan eingebaut werden, welche wiederum Planarbeit für die Erfüllung deren Vorbedingungen mit sich bringen. Diese Heuristik führt allerdings nicht immer zum gewünschten Ergebnis.

Der Planungsprozeß ist erst dann abgeschlossen, wenn sämtliche Ziele, also (initiale) Zielbedingungen und (im Laufe der Planung neu) hinzugekommene Teilziele erfüllt sind.

Die vier Standard-Blockwelt-Operatoren können somit wie folgt definiert werden:

```
Operator:          PICKUP (x)
Vorbedingungen:    x steht auf dem Tisch,
                   x ist frei,
                   Hand hält nichts;
Nachbedingungen:   Dazu-Liste:  Hand hält x,
                   Weg-Liste:   x steht auf dem Tisch,
                                x ist frei,
                                Hand hält nichts.

Operator:          PUTDOWN (x)
Vorbedingungen:    Hand hält x;
Nachbedingungen:   Dazu-Liste:  x steht auf dem Tisch,
                                x ist frei,
                                Hand hält nichts,
                   Weg-Liste:   Hand hält x.

Operator:          STACK (x,y)
Vorbedingungen:    y ist frei,
                   Hand hält x;
Nachbedingungen:   Dazu-Liste:  Hand hält nichts,
                                x steht auf y,
                                x ist frei,
                   Weg-Liste:   y ist frei,
                                Hand hält x.

Operator:          UNSTACK (x,y)
```

```
Vorbedingungen:   Hand hält nichts,
                  x ist frei,
                  x steht auf y;
Nachbedingungen:  Dazu-Liste:  Hand hält x,
                               y ist frei,
                  Weg-Liste:   Hand hält nichts,
                               x ist frei,
                               x steht auf y.
```

Noch eine kurze Bemerkung zur Bezeichnung: Kleinbuchstaben des hinteren Teils des Alphabets (z.B. x,y,z) bezeichnen hier Variablen, Großbuchstaben des vorderen Alphabetblocks (z.B. A,B,C) stehen für Konstanten, welche ganz bestimmte Blöcke repräsentieren.

2.1.2 Rückwärtssuche im Zustandsraum

Es liegt eigentlich nahe, bei der Erstellung eines Planes von der Startsituation des Problems bzw. des Plans auszugehen, indem sukzessive nach Operatoren gesucht wird, deren Vorbedingungen bereits erfüllt sind, und zwar solange, bis sämtliche zu erfüllenden Ziele in irgendwelchen Dazu-Listen von Operatoren auftauchen und deren Gültigkeit bis zur Finalsituation bestehen bleibt.

Daß beim Planen als Suche eine Rückwärtssuche von Ziel- zu Startsituation vorteilhafter ist, liegt nun in einer automatischen Zielzentriertheit, die diese Vorgehensweise mit sich bringt. Der wesentliche Grund dafür ist, daß eine Problemstellung in aller Regel genau eine Start-, aber viele mögliche Zielsituationen spezifiziert, denn die zu erbringenden Ziele beschreiben meist nur einen bestimmten Teilaspekt einer möglichen Finalsituation und zwar den, auf den es ankommt. Aussagen über irgendwelche anderen nicht in einem Ziel eingebundenen Objekte bzw. Fakten werden nicht gemacht.

Zur Erläuterung sei wieder ein Beispiel aus der Blockwelt angeführt: Angenommen, eine Zielsituation sei derart definiert, daß Klotz A auf Klotz B stehen soll, ansonsten aber keine Merkmale spezifiziert werden, die in der erwünschten Finalsituation erfüllt sein sollen. In einer Blockwelt mit weiteren Klötzen als nur A und B, bildet dann jede Situation, in der A auf B steht, eine Zielsituation, unabhängig davon, welche Prädikate für die übrigen Klötze gelten. Gesucht ist also der Weg von einer eindeutig bestimmten Startsituation in eine Klasse von Zielsituationen, wobei letztere durch die Zielmerkmale beschrieben sind.

Daher ist es naheliegend, von diesen wesentlichen Merkmalen der Zielsituation auszugehen, sukzessive Operatoren anzuwenden, in deren erzeugenden Nachbedingungen (add list) eben diese auftauchen, und dann als sekundäre Planziele (= neue Teilziele) die Vorbedingungen der angewandten Operatoren durch weitere Planschritte zu erfüllen.

2.1.3 Mittel-Ziel-Analyse

Das am Ende des vorigen Abschnitts angesprochene Vorgehen bezeichnet man als Mittel-Ziel-Analyse (means ends analysis): Um ein Merkmal M zu erreichen, werden die Vorbedingungen eines Operators X, der M erzeugt, erfüllt und dann X angewandt.

Der Suchraum wird auf diejenigen Situationen eingeschränkt, die potentielle Kandidaten im Rahmen der Operatoranwendungen zur Erfüllung der Planziele

darstellen. Dieser Effekt der Zielgerichtetheit, des Beschränkens auf das Wesentliche, wird dadurch noch unterstützt, daß eine Situation meist mehr Nachfolger als Vorgänger hat, weshalb der Suchraum bei einer Vorwärtssuche sehr schnell unüberschaubare Ausmaße annimmt. Diese Schwierigkeit taucht natürlich auch, wenn auch in abgeschwächter Weise, bei der Rückwärtssuche auf.

Um dieses Problem der Komplexität des Zustandsraums noch besser in den Griff zu bekommen und somit zu effizienteren Planern zu gelangen, können eine Reihe weiterer Techniken eingesetzt werden, die in den folgenden Abschnitten kurz und dann im nächsten Kapitel detailliert an einem konkreten Planungsprogramm aufgezeigt werden.

2.2 Nichtlineares Planen contra lineares Planen

Könnte man von der Linearitätsannahme (linear assumption) ausgehen, wäre die Technik des linearen Planens vollkommen ausreichend, zufriedenstellende Ergebnisse zu liefern. Diese Annahme besagt nämlich, daß es zu jedem Problem eine derartige Bearbeitungsreihenfolge der einzelnen zu erbringenden Teilziele gibt, daß nach und nach eben diese Ziele erfüllt werden und diese auch nach Durchlaufen der gesamten Operatorsequenz ihre Gültigkeit nicht verlieren; bei einer bestimmten Linearisierung der resultierenden Teilpläne gibt es also keine störenden Interaktionen zwischen denselben. Darüber hinaus geht man bei der linear assumption von der weiteren Annahme aus, daß durch Anwendung eines Operators maximal ein Ziel, aber keine Konjunktion mehrerer Teilziele erreicht werden kann.

Auf dieser Linearitätsannahme aufbauend kann man beim linearen Planen gemäß der Mittel-Ziel-Analyse vorgehen, wie dies z.B. bei STRIPS, einem der ersten klassischen Planer, der Fall ist. Dort wird als Suchstrategie durch den Zustandsraum die Tiefensuche gewählt. Die jeweils im Fortschreiten der Planung zu erfüllenden Ziele (= Merkmale der Finalsituation und Vorbedingungen der angewandten Operatoren) werden mit Hilfe eines Kellers verwaltet.

Linearitätsannahme zu unrealistisch

Da sich aber leider obige Voraussetzungen in der Regel als zu idealistisch herausstellen, können sich bei dieser Vorgehensweise häufig Probleme ergeben. Oftmals lassen sich Interaktionen, also Abhängigkeiten oder Wechselwirkungen zwischen den einzelnen Teilplänen (subgoal interactions), nicht durch eine bestimmte Linearisierung der Teilpläne beheben. Es bedarf vielmehr eines "Verzahnens" der jeweiligen Planabschnitte, um zu verhindern, daß in dem einen Planast erfüllte Ziele in einem nachfolgenden wieder zerstört werden. Das läßt sich dadurch in den Planerstellungsprozeß einbauen, daß nachfolgende Teilpläne nicht nur unmittelbar hinter, sondern auch an beliebiger Stelle innerhalb von vorausgegangenen Planabschnitten eingefügt werden können. Es muß dann natürlich wieder gewährleistet sein, daß der eingefügte Teilplan die im nachfolgenden Planabschnitt benötigten und im vorausgegangenen Planabschnitt bereits erzeugten Merkmale nicht zerstört, was durch geeignete Wahl der Einfügestelle zu vermeiden versucht werden muß.

Häufig bedarf es allerdings zur Konfliktlösung einer richtigen Verzahnung der einzelnen Operatoren von Teilplänen und nicht nur eines Ineinanderschachtelns von Teilplänen als Ganze. Doch dies ist bei diesem Ansatz nicht möglich.

Wie man sieht, ist die grundlegende Strategie beim linearen Planen dergestalt, daß eine geordnete Operatorsequenz bestimmt wird, die die gestellten Forderungen erfüllen soll, es wird also eine totale Ordnung der Operatoren erzeugt. Gelingt dies zunächst nicht, wird via Backtracking die feste Reihenfolge modifiziert.

Auftretende Interaktionen machen Mechanismen notwendig, solche Operatorlinearisierungen zu ermitteln, die die Konflikte lösen.

Angebrachter erscheint dann doch die Vorgehensweise, die einzelnen Operatoren im Plan nicht gleich in ein strenges Ordnungsgefüge einzubetten, sondern zunächst nur eine partielle Ordnung von Planschritten zu erzeugen und die bestehende Ordnung erst dann zu verschärfen, wenn Teilplan-Interaktionen dies notwendig machen.

Nichtlineares Planen stellt ein derartiges mächtiges Planungsverfahren dar, das die unnötige Überbeschränkung der Operatorordnung, wie dies beim linearen Planen der Fall ist, vermeidet. Man muß aber an dieser Stelle auch betonen, daß der nichtlineare Planungsansatz nicht aus sich heraus besser ist als der lineare. Die (Un-)Nützlichkeit erweist sich erst im Zusammenspiel mit dem jeweils betrachteten Problembereich.

Der nun folgende Abschnitt soll die Vorteile des nichtlinearen Planens gegenüber dem linearen aufzeigen, bevor dann im nächsten Kapitel an Hand der Beschreibung von TWEAK etwas detaillierter auf diese Thematik eingegangen wird.

2.3 Idee des nichtlinearen Planes

Der entscheidende Unterschied zwischen linearem und nichtlinearem Planen besteht darin, zu welchem Zeitpunkt und in welchem Ausmaß Reihenfolgefestlegungen bezüglich durchzuführender Aktionen getroffen werden.

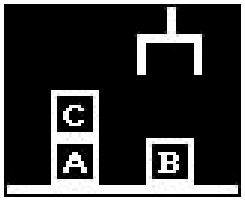
Beim linearen Planen wird eine totale Operator-Ordnung erzeugt und dann zur Behebung von Konflikten eine Umordnung der Operatoren durchgeführt. Naheliegender ist dann doch der Ansatz, zunächst von einer partiellen Operator-Ordnung auszugehen und erst im Bedarfsfall Ordnungsverschärfungen zu vollziehen; dann allerdings zu einem Zeitpunkt, zu dem mehr Fakten vorliegen und daher "bessere" Entscheidungen getroffen werden können. Verfrühte Fehlentscheidungen, die Backtracking nach sich ziehen, können dadurch eher vermieden werden.

Die grundlegende Idee beim nichtlinearen Planen ist also die folgende: Teilziele, die im Laufe der Plangenerierung gleichzeitig an einer Stelle im Plan zu erfüllen sind, werden unabhängig voneinander bearbeitet. Bezüglich der Reihenfolge der daraus resultierenden Teilpläne werden nur dann Festlegungen getroffen, wenn Interaktionen zwischen denselben es notwendig machen.

2.3.1 Das Sussman-Problem

Ein Beispiel aus der Blockwelt, bekannt als die sog. "Sussman-Anomalie", soll zur Erläuterung dienen:

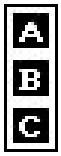
Das Problem, für dessen Lösung ein Plan generiert werden soll, sei durch die Startsituation (Abb. 2.3-1)



- Hand hält nichts,
- C steht auf A,
- B steht auf Tisch,
- A steht auf Tisch,
- C ist frei.

Abb. 2.3-1 Startsituation des Sussman-Problems

und die Zielsituation (Abb. 2.3-2)



- A steht auf B
  (Teilziel 1)
- B steht auf C
  (Teilziel 2)

Abb. 2.3-2 Zielsituation des Sussman-Problems

gegeben.

Die Lösung von Teilziel 1 lautet:
- nehme C von A mit dem Greifarm herunter
  [UNSTACK (C, A)],
- lege C auf den Tisch
  [PUTDOWN (C)],
- ergreife A mit dem Greifarm
  [PICKUP (A)],
- setze A auf B
  [STACK (A, B)].

Die Lösung von Teilziel 2 lautet:
- ergreife B mit dem Greifarm
  [PICKUP (B)],
- setze B auf C
  [STACK (B, C)].

Eine Linearisierung derart, daß man beide resultierenden Teilpläne einfach aneinanderreiht, egal in welcher Reihenfolge, führt zu einem unausführbaren Plan, denn um einen Teilplan ausführen zu können, müssen genau die Ergebnisse des jeweils anderen vorher rückgängig gemacht werden. Ein Verzahnen der beiden Teilpläne ist also erforderlich.

Beim nichtlinearen Planungsansatz werden die beiden Teilziele 1 und 2 zunächst unabhängig voneinander bearbeitet. Die auftretenden Interaktionen zwingen dann zu bestimmten Reihenfolgefestlegungen der beteiligten Planschritte, bis schließlich der folgende (lineare) Plan resultiert:

- nehme C von A mit dem Greifarm herunter
  [UNSTACK (C, A)],
- lege C auf den Tisch
  [PUTDOWN (C)],
- ergreife B mit dem Greifarm
  [PICKUP (B)],
- setze B auf C
  [STACK (B, C)],
- ergreife A mit dem Greifarm
  [PICKUP (A)],
- setze A auf B
  [STACK (A, B)].

Im letzten Kapitel wird die Vorgehensweise des im Rahmen dieser Diplomarbeit implementierten Planungssystems u.a. an Hand dieses Beispiels (mit leicht modifizierten Operatoren zur besseren Übersichtlichkeit) erläutert.

Das Ergebnis eines nichtlinearen Planungsprozesses kann ein vollständig linearisierter Plan sein (siehe etwa obiges Beispiel). Doch in der Regel resultiert daraus ein Plan, der gewisse Freiheiten bezüglich der Operator-Ordnung in sich birgt: ein nichtlinearer Plan. Ein solcher Plan erlaubt eine kompakte Repräsentation insofern, daß er eine Menge von linearen Plänen darstellt, die sich alle durch Vervollständigung, d.h. Linearisierung, aus eben diesem Plan ergeben. Es ist offensichtlich, daß dadurch der Suchraum besser beherrschbar wird, die Menge der möglichen Pläne wird kleiner.

Die Vorzüge des nichtlinearen Planens legen die Vermutung nahe, daß diese Technik gerade dann zu bevorzugen ist, wenn Interaktionen zu erwarten sind. Die eigentliche Zielsetzung dieses Planungsverfahrens besteht allerdings darin, daß mehrere gleichzeitig zu erfüllende Planziele (conjunctive goals) und die mit diesen verbundene Teilpläne unabhängig voneinander bearbeitet werden können. Diese Flexibilität wird erst dann teilweise eingeschränkt, wenn Wechselwirkungen zwischen den einzelnen Planteilen zu Tage treten. Das Planungsmotto lautet also: so wenig Ordnung wie möglich, aber so viel wie nötig.

2.3.2 Interpretation der Nichtlinearität

Ein initialer nichtlinearer Plan könnte folgende Gestalt aufweisen (Abb. 2.3-3):

Abb. 2.3-3 Initialer nichtlinearer Plan

Hierbei stellt der START-Knoten einen virtuellen Operator dar, dessen Vorbedingungen immer erfüllt und dessen Effekte durch die Merkmale der Startsituation beschrieben sind. Der ZIEL-Knoten auf der anderen Seite repräsentiert einen ebenfalls virtuellen Operator, dessen Vorbedingungen es zu erreichen gilt, die primären Planziele also darstellen, welche mit der Finalsituation durch die Problemstellung spezifiziert sind. Die Effekte des ZIEL-Operators sind nicht von Interesse. Der Pfeil zwischen beiden Knoten drückt die Vorgängerbeziehung aus. START- und ZIEL-Operator bilden einen Rahmenplan mit noch unerfüllten Zielen.

Ein möglicher unvollständiger nichtlinearer Plan zur Lösung eines einfachen Blockwelt-Problems könnte wie folgt aussehen (Abb. 2.3-4):

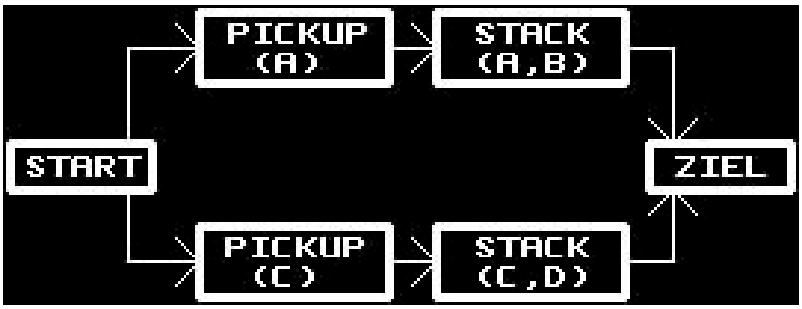


Abb. 2.3-4 Nichtlinearer Plan zur Lösung eines Blockwelt-Problems

Der als gerichteter Graph dargestellte Plan ist noch unvollständig insofern, daß auf Grund noch bestehender Interaktionen zwischen den beiden Planästen eine Verschärfung der Operator-Ordnung vorgenommen werden müßte, um einen Lösungs-Plan zu erhalten, durch dessen Ausführung das gestellte Problem, das an dieser Stelle nicht interessieren soll, gelöst wird.

Was drückt aber ein solcher Plan aus? Die Knoten repräsentieren Operatoren, die gerichteten Kanten Reihenfolgebeziehungen, also die Operator-Ordnung. Doch welche Verzahnung besteht zwischen den parallelen Planästen? Es sind verschiedene Interpretationen möglich. Für uns soll die Festlegung gelten, daß die Operatoren paralleler Zweige in jeder totalen Ordnung ausgeführt werden dürfen, die mit der partiellen kompatibel ist. Somit repräsentiert obiger Plan (Abb. 2.3-4) sämtliche folgenden linearen Operatorsequenzen bzw. Pläne:

- PICKUP (A), STACK (A, B), PICKUP (C) , STACK (C, D) ;
- PICKUP (A), PICKUP (C) , STACK (C, D), STACK (A, B) ;
- PICKUP (A), PICKUP (C) , STACK (A, B), STACK (C, D) ;
- PICKUP (C), STACK (C, D), PICKUP (A) , STACK (A, B) ;
- PICKUP (C), PICKUP (A) , STACK (C, D), STACK (A, B) ;
- PICKUP (C), PICKUP (A) , STACK (A, B), STACK (C, D) .

Nach welchem Algorithmus nun ein nichtlinearer Plan aus einer Beschreibung der verfügbaren Operatoren, der gegebenen Start- und erwünschten Zielsituation gewonnen werden kann, soll an Hand eines konkreten Planers, TWEAK, im nachfolgenden Kapitel aufgezeigt werden.

```
============================================================
Kapitel 3:    TWEAK -
              ein nichtlinearer domänen-unabhängiger Planer
============================================================
```

Diesem Kapitel liegt ein Artikel [Chapman 1987] über TWEAK zugrunde, den der Ersteller dieses Planungssystems, Chapman, selbst verfaßt hat. Darin bezeichnet er TWEAK als eine "strenge mathematische Rekonstruktion" vorausgegangener nichtlinearer domänen-unabhängiger Planungsprogramme. Inwiefern dies zutrifft und ob ein derartiger Ansatz den bis daher eher heuristisch und daher schwerer nachvollziehbar, aber in begrenztem Umfang erfolgreich arbeitenden Planern überlegen ist, soll in diesem Kapitel kritisch betrachtet werden.

Im Gegensatz zu Planungssystemen wie z.B. MOLGEN, bei denen auf mehreren Abstraktionsstufen geplant wird, findet bei TWEAK wie auch bei anderen rein nichtlinearen Planern der eigentliche Planungsprozeß auf einer Ebene statt. Dennoch lassen sich auch hierbei verschiedene zentrale Aktivitäten unterscheiden, deren Anstoß lediglich von einer übergeordneten Kontrollebene erfolgt, die gewissermaßen die Regie über den Planungsprozeß führt insofern, als daß das jeweilig aktuell zu bearbeitende Planziel hier ausgewählt wird.

Bevor ab Abschnitt 3.2 auf die obig angesprochenen "zentralen Aktivitäten" näher eingegangen wird, seien zunächst noch in Anlehnung an die Einführung in das nichtlineare Planen (siehe Kapitel 2) die bei TWEAK verwendeten Darstellungskonventionen bzw. -restriktionen aufgeführt, um dann langsam, aber sicher in medias res zu kommen.

3.1 Komponenten eines TWEAK-Plans

Die "Bausteine", aus denen TWEAK einen nichtlinearen Plan zusammensetzt, sind zum einen die zur Verfügung stehenden Operatoren (genauer: Operator-Schablonen), zum andern sog. Constraints, welche in die an sich ungeordnete Menge der angewandten Operatoren (Operator-Instanzen) eine gewisse Ordnung bringen.

Dem Planer werden zur Lösung eines Problems Operator-Schablonen verfügbar gemacht, welche im Falle der Anwendbarkeit durch Einbau in den gerade bearbeiteten Plan mit dem Zweck der Erfüllung eines Teilzieles zu Operator-Instanzen werden mit Bezügen zu den anderen Operator-Instanzen im Plan via Constraints.

3.1.1 Operatoren

Wie auch bei anderen Planungssystemen besteht bei TWEAK ein Operator aus einer endlichen Menge von Merkmalen - in [Chapman 1987] als propositions bezeichnet -, genauer gesagt aus endlich vielen Vor- und Nachbedingungen. Letztere sind hier aber nun nicht mehr explizit in Dazu- und Wegliste untergliedert, vielmehr tauchen die Merkmale, die ein Operator zerstört oder löscht, in seinen Nachbedingungen in negierter Form auf, und diejenigen, die er erzeugt, in nichtnegierter Form.

Hierbei ist noch anzumerken, daß ein Unterschied besteht zwischen der Erzeugung eines negierten Merkmals (z.B. Erzeugung von (NICHT (ist_frei A))) und der Zerstörung des nichtnegierten Parts eben dieses Merkmals (z.B. Zerstörung von (ist_frei A)).

Die Negiertheit der löschenden Effekte soll bei TWEAK lediglich die Zugehörigkeit der betreffenden Nachbedingung zur delete list des betrachteten Operators anzeigen, dient also gewissermaßen als Markierung und darf nicht so interpretiert werden, daß damit eine Erzeugung des Negats erfolgen soll. Diese Darstellung ist zwar etwas irreführend, verhindert aber die Notwendigkeit, die Nachbedingungen eines Operators in Dazu- und Wegliste zu untergliedern, was wohl ausschlaggebend dafür war, daß Chapman diese Repräsentation wählte und auch von mir bei dem im Rahmen dieser Diplomarbeit implementierten Planer (siehe Kapitel 4) übernommen wurde.

Die Erzeugung negierter Merkmale im eigentlichen Sinne ist auf Grund dieser Konvention nicht darstellbar, aber dazu besteht auch keine Notwendigkeit, da bei der gewählten Aktionsrepräsentation negierte Vorbedingungen als zu erfüllende Teilziele nicht möglich sind.

Beispiel:

Operator: **PICKUP** (x)

Vorbedingungen:
(steht_auf_Tisch x),
(ist_frei x),
(Hand_hält nichts);

Nachbedingungen:
(Hand_hält x),
(NICHT (steht_auf_Tisch x)),
(NICHT (ist_frei x)),
(NICHT (Hand_hält nichts)).

Für Merkmale wird die funktionale Schreibweise gewählt, d.h. sie werden als Listen repräsentiert, deren erstes Element ein Prädikat bezeichnet; die nachfolgenden Elemente, welche Variablen oder Konstanten sein können, stellen diesbezüglich die Argumente dar. Alle Merkmale zeichnen sich durch lediglich atomaren Aufbau aus, d.h. Funktionen, logische Verknüpfungen und Quantifizierungen bezüglich der Argumente sind nicht zulässig.

Gründe für diese sehr eingeschränkte Aktionsrepräsentation sind darin zu sehen, daß TWEAK den Anspruch erhebt, korrekt und vollständig bei der Planerstellung vorzugehen. Mächtigere Darstellungen könnten Nebeneffekte zulassen, so daß nicht mehr alle Änderungen der Welt explizit in den Nachbedingungen der Operatoren formuliert werden müßten. Darüber hinaus könnten Operator-Anwendungen von der jeweiligen Eingabesituation abhängen und auf diese Weise Effekte ermöglichen, die nur in bestimmten Situationen erzielt werden sollen; auch dadurch wäre flexiblere Aktionsplanung möglich. Warum durch solche Maßnahmen die Korrektheit und/oder Vollständigkeit aufgegeben und deshalb lieber derartige Restriktionen in Kauf genommen werden, so daß dabei ein praktisch kaum einsetzbarer Planer resultiert, wird erst klar, wenn die Vorgehensweise TWEAKs näher erläutert worden ist. Die Antwort auf diese Frage sei daher auf das Ende dieses Kapitels verlagert.

3.1.2 Constraints

Wie bereits im vorigen Kapitel erwähnt, bildet ein initialer Plan (Abb. 2.3-3) den Ausgangspunkt der Planerstellung. Es gilt hierbei, diesen unfertigen Plan zu vervollständigen, bis schließlich ein (nicht-)linearer Plan resultiert, der sämtliche geforderten Ziele erfüllt. Genauer muß man sagen, daß der Planungsprozeß dann abgeschlossen werden kann, wenn alle Vervollständigungen, d.h alle möglichen Linearisierungen und Instantiierungen des bis dahin erstellten nichtlinearen und unvollständig spezifizierten Plans, das gegebene Problem lösen.

Wie läßt sich ein Plan vervollständigen? Zum einen natürlich dadurch, daß man Operatoren integriert, die bestimmte Effekte bewirken sollen. Darüber hinaus ist es aber notwendig, die Beziehungen, die zwischen den einzelnen Operatoren bestehen müssen, auszudrücken. Erst dadurch entsteht aus der ungeordneten Menge eingebauter Operatoren ein Plan-Gefüge. Derartige Beziehungen können beispielsweise wie folgt aussehen:

- Operator O1 muß vor Operator O2 angewandt werden
  (Abb. 3.1-1, Abb. 3.1-2):

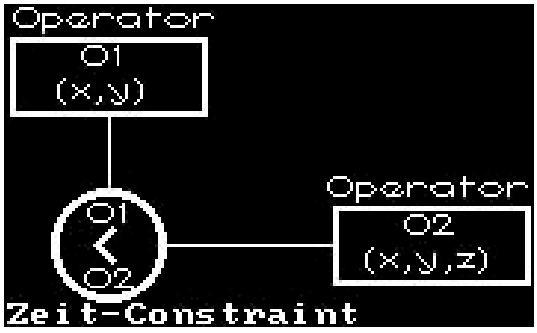


Abb. 3.1-1 Beispiel eines Zeit-Constraints: O1 < O2

Obige Beziehung zwischen O1 und O2 (Abb. 3.1-1) kann auch wie folgt kürzer dargestellt werden (Abb. 3.1-2):

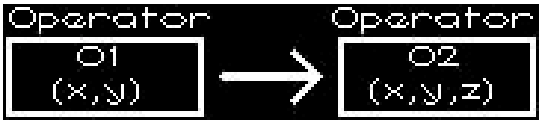


Abb. 3.1-2 Verkürzte Darstellung des Zeit-Constraints O1 < O2

- Operator O1 muß bezüglich der ersten Argumentposition
  (hier: x) auf dasselbe Argument wie Operator O2 bezüglich der dritten
  Argumentposition (hier: z) angewandt werden (Abb. 3.1-3):

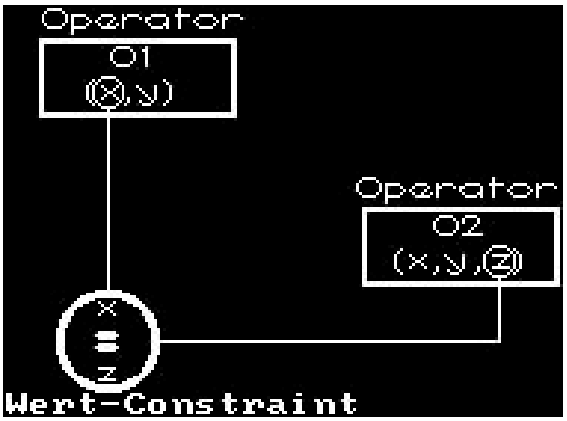


Abb. 3.1-3 Beispiel eines Wert-Constraints: x [Operator O1] = z [Operator O2]

- Operator O1 darf bezüglich der ersten Argumentposition (hier: x) nicht auf dasselbe Argument wie Operator O2 bezüglich der dritten Argumentposition (hier: z) angewandt werden (Abb. 3.1-4):

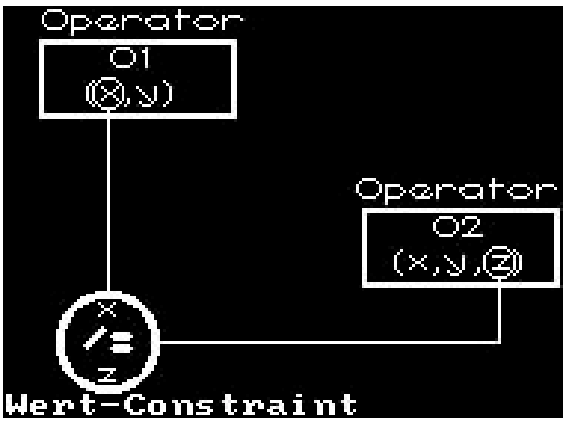


Abb. 3.1-4 Beispiel eines Wert-Constraints: x [Operator O1] /= z [Operator O2]

- Operator O muß bezüglich der ersten Argumentposition (hier: x) auf ein konstantes Argument (hier: B) angewandt werden (Abb. 3.1-5, Abb. 3.1-6):

Abb. 3.1-5 Verkürzte Darstellung des Wert-Constraints: x [Operator O] = B

Die Instantiierung der ersten Variable des Operators O in obiger Darstellung (Abb. 3.1-5), realisiert durch ein entsprechendes Wert-Constraint, wird deutlicher durch folgende ausführlichere Darstellung (Abb. 3.1-6):

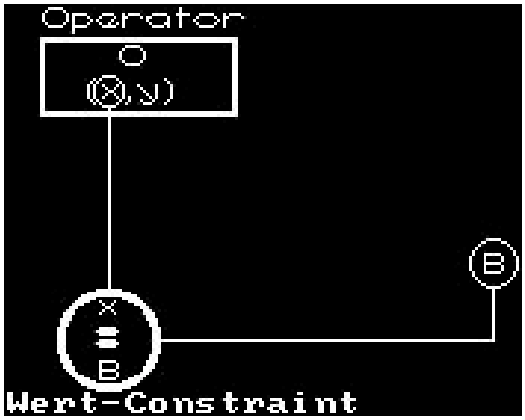


Abb. 3.1-6 Darstellung des Wert-Constraints x [Operator O] = B

Bei TWEAK wird durch die Verwendung von Constraints nichtlineares Planen ermöglicht. Hierbei sind zweierlei Arten von Constraints im Spiel:

Def. 3.1-1:

(1) Zeit-Constraints (temporal constraints) drücken Reihenfolgebeziehungen zwischen den einzelnen Operatoren aus, sie auferlegen dem Plan also eine (partielle) Ordnung:

O1 < O2 (O1, O2 seien beliebige Operator-Instanzen; siehe auch Abb. 3.1-1, 3.1-2)
{ in Worten:
O1 steht in direkter Zeit-Relation zu O2. }
:<=> Operator-Instanz O1 ist im Plan direkter Vorgänger von Operator-Instanz O2.
Anm.: - Die Zeit-Relation ist weder reflexiv noch symmetrisch, aber transitiv.

O1 $<_n$ O (O1, O seien wiederum beliebige Operator-Instanzen; n sei eine natürliche Zahl > 1.)
{ in Worten:
O1 steht in Zeit-Relation zu O. }
:<=> Operator-Instanz O1 ist im Plan (nicht direkter) Vorgänger von Operator-Instanz O.
Anm.: - "$<_1$" ist gleichbedeutend mit "<".

(2) Wert-Constraints ((non-)codesignation constraints) dienen dazu, die einzelnen Planschritte (= Operator-Instanzen) genauer zu spezifizieren; die möglichen Instantiierungswerte der in Merkmalen enthaltenen Variablen werden eingegrenzt. Diese Constraints stellen nichts anderes als *Unifikations-Constraints* zwischen den gebundenen Merkmalen dar. Zu beachten ist hierbei, daß Wert-Constraints Merkmale und nicht Argumente explizit verbinden; die Wert-Bindungen zwischen den betreffenden Merkmal-Argumenten, angedeutet in den Abbildungen Abb. 3.1-3 bis 3.1-6, ergeben sich daraus implizit. Zwischen Argumenten - Variablen und Konstanten - sind damit folgende Beziehungen möglich:

```
(i) x = z (x, z seien beliebige Variablen;
           x sei Argument eines Operators O1,
              z sei Argument eines Operators O2;
              siehe auch Abb. 3.1-3)
{ in Worten:
    x steht in direkter Gleichheits-Wert-Relation (Codesignations-
    Relation) zu z. }
:<=>  Variable x (lokal zu Operator O1) soll mit demselben Wert
      instantiiert werden wie Variable z (lokal zu Operator O2).
      Die beiden Variablen x und z sind explizit via Wert-Constraint,
      genauer: Codesignation-Wert-Constraint, verbunden.
Anm.: - O.B.d.A. gelte, daß O1 und O2 verschieden seien.
        Falls O1 und O2 identisch sind, kann auf das Setzen eines
        entsprechenden Constraints zwischen x und z verzichtet werden,
        indem die beiden Variablen denselben Namen erhalten; die Wirkung
        ist dieselbe.
      - Die Bezeichnungen von Variablen verschiedener Operator-Instanzen
        haben keinerlei Auswirkungen aufeinander.
        Sollen zwei Variablen verschiedener Operator-Instanzen mit
        denselben Werten instantiiert werden, so muß dies durch Setzen
        eines entsprechenden Wert-Constraints explizit oder implizit über
        Hüllenbildung auch bei identischen Variablennamen vermerkt werden.
      - Die Gleichheits-Wert-Relation ist reflexiv, symmetrisch und
        transitiv; sie stellt somit eine Äquivalenzrelation dar.

(ii) x /= z (x, z seien beliebige Variablen;
             x sei Argument eines Operators O1,
             z sei Argument eines Operators O2;
             siehe auch Abb. 3.1-4)
{ in Worten:
    x steht in direkter Ungleichheits-Wert-Relation (Noncodesignations-
    Relation) zu z. }
:<=>  Variable x (lokal zu Operator O1) darf nicht mit demselben Wert
      instantiiert werden wie Variable z (lokal zu Operator O2).
      Die beiden Variablen x und z sind explizit via Wert-Constraint,
      genauer: Noncodesignation-Wert-Constraint, verbunden.
Anm.: - Die Bezeichnungen von Variablen haben bei dieser Ausprägung der
        Wert-Relation keinerlei Auswirkungen aufeinander; dabei spielt es
        keine Rolle, ob die betrachteten Variablen Argumente ein und
        derselben Operator-Instanz sind oder ob sie sich auf verschiedene
        Operator-Instanzen beziehen.
      - Die Ungleichheits-Wert-Relation ist weder reflexiv noch transitiv,
        aber symmetrisch.

(iii) x = B (x sei eine beliebige Variable;
             x sei Argument eines Operators O;
             B sei eine Konstante;
             siehe auch Abb. 3.1-5 und 3.1-6)
{ in Worten:
    x steht in direkter Gleichheits-Wert-Relation zu B. }
:<=>  Variable x (lokal zu Operator O) soll an die Konstante B gebunden
      werden, was einer Instantiierung mit B gleichkommt.
      Variable x ist mit Konstante B explizit via (Codesignation-)Wert-
      Constraint verbunden.

(iv) x /= B (x sei eine beliebige Variable;
             x sei Argument eines Operators O;
             B sei eine Konstante; o. Abb.)
{ in Worten:
    x steht in direkter Ungleichheits-Wert-Relation zu B. }
:<=>  Variable x (lokal zu Operator O) darf nicht an die Konstante B
      gebunden werden.
```

```
      Variable x ist mit Konstante B explizit via (Noncodesignation-)Wert-
      Constraint verbunden.

(v) x =n y (x, y seien beliebige Variablen;
            x sei Argument eines Operators O1,
            y sei Argument eines Operators O2;
            n sei eine natürliche Zahl > 1.)
{ in Worten:
    x steht in Gleichheits-Wert-Relation zu y. }
:<=>  Variable x (lokal zu Operator O1) soll mit demselben Wert
      instantiiert werden wie Variable y (lokal zu Operator O2).
      Die beiden Variablen x und y sind nicht explizit via Wert-Constraint
      verbunden, sondern die Bindung ergibt sich über die transitive Hülle
      der Wert-Relation.
Anm.: - siehe (i).
      - "=1" ist gleichbedeutend mit "=".

(vi) x /=n y (x, y seien beliebige Variablen;
              x sei Argument eines Operators O1,
              y sei Argument eines Operators O2;
              n sei eine natürliche Zahl > 1.)
{ in Worten:
    x steht in Ungleichheits-Wert-Relation zu y.}
:<=>  Variable x (lokal zu Operator O1) darf nicht mit demselben Wert
      instantiiert werden wie Variable y (lokal zu Operator O2).
      Die beiden Variablen x und y sind nicht explizit via Wert-Constraint
      verbunden, sondern die Bindung ergibt sich über die transitive Hülle
      der Wert-Relation.
Anm.: - siehe (ii)
      - "/=1" ist gleichbedeutend mit "/=".

(vii) x =n A (x sei eine beliebige Variable;
              x sei Argument eines Operators O;
              A sei eine Konstante;
              n sei eine natürliche Zahl > 1.)

{ in Worten:
    x steht in Gleichheits-Wert-Relation zu A. }
:<=>  Variable x (lokal zu Operator O) soll an die Konstante A gebunden
      werden, was einer Instantiierung mit A gleichkommt.
      Variable x ist mit Konstante A nicht explizit via Wert-Constraint
      verbunden, sondern die Bindung ergibt sich über die transitive Hülle
      der Wert-Relation.

(viii) x /=n A (x sei eine beliebige Variable;
                x sei Argument eines Operators O;
                A sei eine Konstante;
                n sei seine natürliche Zahl > 1.)
{ in Worten:
    x steht in Ungleichheits-Wert-Relation zu A.}
:<=>  Variable x (lokal zu Operator O) darf nicht an die Konstante A
      gebunden werden.
      Variable x ist mit Konstante A nicht explizit via Wert-Constraint
      verbunden, sondern die Bindung ergibt sich über die transitive Hülle
      der Wert-Relation.
```

3.2 Planen mit Constraints

An Hand des folgenden Ausschnitts eines nichtlinearen Plans (Abb. 3.2-1) soll der Gebrauch von Constraints in TWEAK noch einmal veranschaulicht werden:

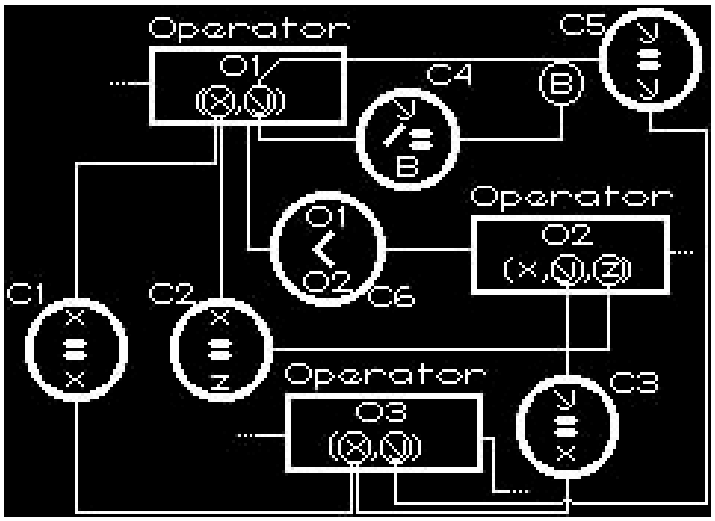


Abb. 3.2-1 Ausschnitt eines nichtlinearen Plans

Zur Veranschaulichung des Gebrauchs von Constraints
(C1-C3, C5: Codesignation-Wert-Constraints;
C4: Noncodesignation-Wert-Constraint;
C6: Zeit-Constraint).

Das Setzen von Constraints wird im Englischen als "constraint posting" bezeichnet.

Innerhalb des gegebenen Plan-Ausschnitts (Abb. 3.2-1) haben folgende Relationen Gültigkeit (in den eckigen Klammern hinter den Variablen-Bezeichnungen ist zur eindeutigen Kennzeichnung jeweils der entsprechende Bezugs-Operator angegeben):

x [O1] = x [O3] (über Wert-Constraint C1),
x [O1] = z [O2] (über Wert-Constraint C2),
x [O1] $=_n$ y [O2] (über Wert-Constraints C1,C3; n=2),
y [O1] /= B (über Wert-Constraint C4),
y [O1] = y [O3] (über Wert-Constraint C5),
y [O2] = x [O3] (über Wert-Constraint C3),
y [O2] $=_n$ z [O2] (über Wert-Constraints C3,C1,C2; n=3),
z [O2] $=_n$ x [O3] (über Wert-Constraints C2,C1; n=2),
y [O3] /$=_n$ B (über Wert-Constraints C5,C4; n=2),

O1 < O2 (über Zeit-Constraint C6).

Def. 3.2-1: Seien v, w beliebige Argumente von Operator-Instanzen, also Variablen oder Konstanten, und n eine natürliche Zahl:

(1) Argument-Codesignation:
v codesigniert mit w,
in Zeichen: v $=^!$ w
:<=> v $=_n$ w für n > 0.

(2) Argument-Noncodesignation:
v noncodesigniert mit w,
in Zeichen: v /$=^!$ w

:<=> v /=$_n$ w für n > 0.

(3) Argument-Wert-Relation:
v steht in Wert-Relation zu w,
in Zeichen: v <-> w
:<=> v =$_n$ w oder v /=$_n$ w
für n > 0.

Anm.: Für zwei verschiedene Konstanten A und B gilt immer:
- A =$^!$ A,
- A /=$^!$ B.
Zwei verschiedene Konstanten können also nie zueiander in Codesignations-Relation stehen!

Mit dieser Definition ergeben sich im obigen Plan-Ausschnitt (Abb. 3.2-1) folgende Beziehungen:

- x [O1] codesigniert mit: x [O3], z [O2], y [O2] und
  noncodesigniert mit: --- ;
- y [O1] codesigniert mit: y [O3] und
  noncodesigniert mit: B;

- x [O2] codesigniert mit: --- und
  noncodesigniert mit: --- ;
- y [O2] codesigniert mit: x [O3], x [O1], z [O2] und
  noncodesigniert mit: --- ;
- z [O2] codesigniert mit: x [O1], x [O3], y [O2] und
  noncodesigniert mit: --- ;

- x [O3] codesigniert mit: x [O1], y [O2], z [O2] und
  noncodesigniert mit: --- ;
- y [O3] codesigniert mit: y [O1] und
  noncodesigniert mit: B.

Bei TWEAK werden also Constraints eingesetzt, um eben solche "Randbedingungen" zu formulieren, die im Laufe der Plan-Generierung zu beachten sind. Durch das Setzen von Constraints wird der Suchraum sukzessive eingegrenzt, ohne eventuell zu revidierende Festlegungen treffen zu müssen.

Chapman, TWEAKs Implementator, bezeichnet das Setzen von Constraints als den Vorgang des Definierens eines Objektes, in diesem Falle eines Plans, dadurch, daß nach und nach partielle Beschreibungen erstellt werden, die das zu erzeugende Objekt immer detaillierter spezifizieren; die Menge der in Frage kommenden Objekte wird schrittweise eingegrenzt. Bevor konkrete Objekteigenschaften festgelegt werden, wird zunächst möglichst viel Information zusammengetragen, die das gewünschte Verhalten des Objektes beschreiben, um dann zu einem späteren Zeitpunkt im Rahmen der Objektdefinition auf mehr Wissen beruhende Entscheidungen treffen zu können, die mit grösserer Wahrscheinlichkeit zum Ziel führen.

Constraint-Festlegungen erfolgen bei TWEAK sowohl statisch a priori, etwa in Form lokaler Constraints bei den Definitionen der einzelnen Operator-Schablonen oder bei der Reihenfolgefestlegung von Start- und Zieloperator beim Erstellen des Initialplans, wie auch und vor allem dynamisch während des Planerstellungsprozesses.

3.2.1 Exkurs in die Modallogik

Unter Modallogik versteht man den Zweig der formalen Logik, in dem zur Bildung von Aussagen auch die Modalitäten herangezogen werden.

Die partielle Operator-Ordnung in nichtlinearen Plänen, in TWEAK repräsentiert mittels Zeit-Constraints, und die partielle Festlegung der Variablenwerte durch Wert-Constraints lassen gewisse Reihenfolgebeziehungen und Variableninstantiierungen offen oder bieten zumindest die Möglichkeit, sie offen zu lassen. Daraus ergibt sich für die Gültigkeit von Merkmalen innerhalb eines Plans, daß manche Merkmale notwendigerweise, d.h. auf jeden Fall, also bei beliebiger Linearisierung der bestehenden partiellen Operator-Ordnung und beliebiger zu den bestehenden Wert-Constraints konsistenter Instantiierung der verwendeten Variablen gelten bzw. nicht gelten, andere allerdings nur möglicherweise, also bei bestimmter Linearisierung und/oder Instantiierung.

Als Konsequenz läßt sich ableiten, daß eine Operator-Instanz ein Merkmal notwendigerweise oder nur möglicherweise erzeugt oder zerstört, wobei bei der gewählten Aktionsrepräsentation, wie bereits hingewiesen, eine Zerstörung gleichbedeutend ist mit der Erzeugung des entsprechenden löschenden Merkmals. Es sei noch angemerkt, daß eine Operator-Instanz ein Merkmal dann erzeugt/zerstört, wenn eben dieses Merkmal mit einer erzeugenden/löschenden, also nichtnegierten/negierten Nachbedingung des Operators unifiziert werden kann.

Als Schreibweise sei für die weiteren Betrachtungen vereinbart, daß ein nachfolgendes hochgestelltes Fragezeichen ("?"), etwa hinter dem Wert-Relations-Zeichen "$=^{?}$", die Möglichkeit symbolisiere, ein nachfolgendes hochgestelltes Ausrufezeichen ("!") hingegen die Notwendigkeit anzeige.

Auch dies sei wieder an einem Beispiel verdeutlicht (Abb. 3.2-2):

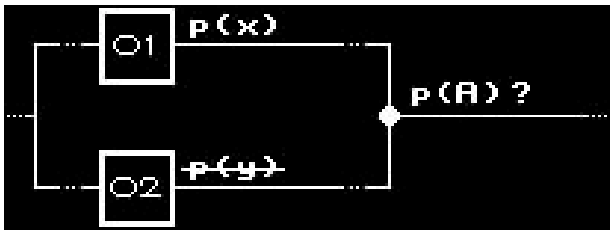


Abb. 3.2-2 Ausschnitt eines nichtlinearen Plans
(die Kanten sollen die Operator-Ordnung andeuten)

Die Operator-Instanz O1 des obigen Plan-Ausschnitts (Abb. 3.2-2) habe u.a. folgende Nachbedingung: p(x) oder in funktionaler Schreibweise (p x), wobei p ein Prädikat-Symbol sei. Die Operator-Instanz O2 des parallelen Plan-Astes verfüge über den löschenden Effekt (NICHT (p y)). Für die Gültigkeit eines Merkmals (p A) an der Stelle im Plan, an der die beiden Äste mit den Operator-Instanzen O1 und O2 rechts zusammenlaufen, kann in diesem Stadium der Plan-Erstellung noch keine "sichere" Aussage getroffen werden: (p A) ist an der betrachteten Stelle möglicherweise erfüllt, d.h. es gibt eine Linearisierung des nichtlinearen Plans und eine Variableninstantiierung, bei der die Gültigkeit sicher gegeben ist; wenn etwa derart linearisiert wird, daß O2 < O1 gilt, und die Argument-Variable x von O1 an die Konstante A gebunden wird, dann gilt das Merkmal (p A) notwendigerweise an der betrachteten Stelle im Plan.

3.2.2 Unifikation von Merkmalen

Codesignation-Wert-Constraints drücken eine Äquivalenzrelation zwischen Variablen bzw. Variablen und Konstanten aus. In einem vollständigen Plan müssen sämtliche Variablen auf diese Weise an Konstanten gebunden sein, so daß bei Ausführung des Plans für die Variablen die spezifischen Konstanten substituiert werden können. Ein von TWEAK erstellter nichtlinearer Plan kann mehrere vollständige Pläne darstellen.

Wert-Constraints können, wie im vorigen Abschnitt erwähnt, auch an ganze Merkmale gebunden sein:

Def. 3.2-2: Seien (p a_1 ... a_k) (=: M1) und (q b_1 ... b_m) (=: M2) beliebige Merkmale, wobei p und q Prädikat-Symbole, a_i [1<=i<=k] und b_j [1<=j<=m] Argumente, k und m natürliche Zahlen > 0 sind.

(1) Merkmal-Codesignation:
M1 codesigniert mit M2,
in Zeichen: M1 $=^!$ M2
:<=> - beide Merkmale sind negiert oder
beide sind nichtnegiert,
und
- die Argument-Anzahl beider Merkmale ist identisch,
d.h. k = m,
und
- beide Merkmale weisen dasselbe Prädikat-Symbol auf,
d.h. p = q,
und
- alle korrespondierenden Argumente beider Merkmale
codesignieren, d.h. $a_i =^! b_i$ für alle i aus [1; k].

(2) Merkmal-Noncodesignation:
M1 noncodesigniert mit M2,
in Zeichen: M1 $/=^!$ M2
:<=> - beide Merkmale sind negiert oder
beide sind nichtnegiert,
und
- die Argument-Anzahl beider Merkmale ist identisch,
d.h. k = m,
und
- beide Merkmale weisen dasselbe Prädikat-Symbol auf,
d.h. p = q,
und
- mindestens ein Paar korrespondierender Argumente beider Merkmale noncodesigniert,
d.h. $a_i /=^! b_i$ für mindestens ein i aus [1; k].

(3) Merkmal-Wert-Relation:
M1 steht in Wert-Relation zu M2,
in Zeichen: M1 <-> M2
:<=> M1 $=^!$ M2 oder M1 $/=^!$ M2.

Will man also zwei Merkmale M1 und M2 per Wert-Constraint aneinander binden, so läuft dies (im Falle M1 = M2) auf einen Unifikationsprozeß hinaus, dessen Ergebnis Wert-Relationen zwischen sämtlichen korrespondierenden Elementen beider Merkmale sind; im Falle M1 /= M2 ist der Constraint-Wirkung dadurch genüge getan, daß (mindestens) zwischen zwei korrespondierenden Elementen von M1 und M2 Argument-Noncodesignation gilt.

Beispiele: (hierbei werden nur Variablenbindungen betrachtet; Linearisierungsbetrachtungen bleiben ausser acht!)

```
1) (on  x y)  =  (on  A z)   <=>   [x  =  A  und  y  =  z]
2) (on  x y) /=  (on  A z)   <=>   [x /=  A  oder y /=  z]
3) (on  A B)  =  (on  A C)  Unifikation  scheitert, da zwei verschiedene
                            Konstanten (hier B und C) nie zueinander in
                            "="-Wert-Relation stehen können!
4) (on  A y)  =  (on  A z)   <=>   y  =  z
5) (on  A y) /=  (on  A z)   <=>   y /=  z
6) (on  A B)  =  (on  A B)   <=>   TRUE
7) (on  A B)  =  (up  A B)  Unifikation scheitert!
                            (gleiche Begründung wie bei 3) bezüglich der
                             Elemente on und up)
8) [x  =  A  und  y, z ungebunden]
                              =>   (on  x y)  =?  (on  A z)
9) [x  =  A  und  y  =  z]    =>   (on  x y)  =!  (on  A z)
```

3.3 Zentrale Planungsaktivitäten bei TWEAK

Während der Erstellung eines Plans stellt sich immer wieder das Problem festzustellen, ob an einer Stelle im Plan ein bestimmtes Merkmal gilt oder nicht. Dies ist beim linearen Planen genauso der Fall, nur daß dort durch die straffere Operator-Ordnung leichter derartige Aussagen getroffen werden können. Variablen und nur partiell linearisierte Operatoren bei TWEAKs Plänen erschweren diesbezüglich die Arbeit: es ist dann nicht nur so, daß bestimmte Sachverhalte gelten und andere nicht, sondern manche Fakten können gelten bzw. nicht gelten, müssen es aber nicht unbedingt, je nachdem, wie der vorliegende Plan vervollständigt wird (siehe Abschnitt 3.2.1). Abhilfe schafft hier ein Kriterium, das sog. Erfüllungskriterium, mit dessen Hilfe mit polynomiellem Aufwand die Gültigkeit eines Merkmals an einer Stelle im Plan festgestellt werden kann.

Die eigentliche Planarbeit besteht darin, solche Planmodifikationen durchzuführen, daß ein auf der übergeordneten Kontrollebene gerade ausgewähltes Ziel erreicht wird. Die Vorgehensweise basiert dabei im wesentlichen auf dem eben erwähnten Kriterium. Ziele sind zum einen die Merkmale der Finalsituation, zum anderen die Vorbedingungen der angewandten Operatoren.

Dieses Kriterium steht auch im Mittelpunkt bei der Konflikt-Auflösung; Näheres dazu siehe Abschnitt 3.3.3.

3.3.1 Das Erfüllungskriterium

Eine Möglichkeit, die "Unsicherheiten" (partielle Operator-Ordnung und partielle Beschreibung der Variablenwerte) in den Griff zu bekommen, bestünde darin, sämtliche Vervollständigungen nacheinander zu betrachten, wenn es darum ginge, die Gültigkeit eines Merkmals in einer bestimmten Situation zu ermitteln. Dies wäre ein möglicher, aber sehr umständlicher Weg, da exponentiell viele vollständige Pläne jeweils durch einen derartigen nichtlinearen Plan repräsentiert werden.

Eine Methode, die mit polynomiellem Aufwand (bezogen auf die Zahl der Planschritte) auskommt, konzentriert sich auf die bezüglich eines betrachteten Merkmals relevanten Komponenten, sprich Operator-Instanzen, eines

unvollständigen Plans und basiert auf einem Sachverhalt, welcher in Abb. 3.3-1 graphisch skizziert wird.

Vorneweg noch einige Definitionen, die eine prägnantere Formulierung des Erfüllungskriteriums erlauben und auch für die nachfolgenden Überlegungen herangezogen werden:

Def. 3.3-1: Seien v, w beliebige Argumente von Operator-Instanzen, also Variablen oder Konstanten, und n eine natürliche Zahl > 0:

(1) Argument-Codesignierbarkeit:
v ist codesignierbar mit w,
in Zeichen: $v =^{?} w$
:<=> - $v =^{!} w$
oder
- es gibt kein n > 0, so daß $v /=_n w$.

(2) Argument-Noncodesignierbarkeit:
v ist noncodesignierbar mit w,
in Zeichen: $v /=^{?} w$
:<=> es gibt kein n > 0, so daß $v =_n w$.

Def. 3.3-2: Seien $(p\ a_1 \ldots a_k)$ (=: M1) und $(q\ b_1 \ldots b_m)$ (=: M2) beliebige Merkmale, wobei p und q Prädikat-Symbole, a_i [1<=i<=k] und b_j [1<=j<=m] Argumente, k und m natürliche Zahlen > 0 sind.

(1) Merkmal-Codesignierbarkeit:
M1 ist codesignierbar mit M2,
in Zeichen: $M1 =^{?} M2$
:<=> - $M1 =^{!} M2$
oder
- beide Merkmale sind negiert oder
beide sind nichtnegiert,
und
die Argument-Anzahl beider Merkmale ist identisch,
d.h. k = m,
und
beide Merkmale weisen dasselbe Prädikat-Symbol auf,
d.h. p = q,
und
alle korrespondierenden Argumente beider Merkmale sind
codesignierbar, d.h. $a_i =^{?} b_i$ für alle i aus [1; k].

(2) Merkmal-Noncodesignierbarkeit:
M1 ist noncodesignierbar mit M2,
in Zeichen: $M1 /=^{?} M2$
:<=> - $M1 /=^{!} M2$
oder
- beide Merkmale sind negiert oder
beide sind nichtnegiert,
und
die Argument-Anzahl beider Merkmale ist identisch,
d.h. k = m,
und
beide Merkmale weisen dasselbe Prädikat-Symbol auf,
d.h. p = q,
und
mindestens ein Paar korrespondierender Argumente beider
Merkmale ist noncodesignierbar, d.h. $a_i /=^{?} b_i$
für mindestens ein i aus [1; k].

Def. 3.3-3: Seien O eine Operator-Instanz und M, M' Merkmale.

(1) O ist ein (etablierter) Erzeuger von M
:<=>
M' ist eine erzeugende, d.h. nichtnegierte Nachbedingung von O,
und es gilt $M =^{!} M'$.

(2) O ist ein potentieller Erzeuger von M
:<=>
M' ist eine erzeugende Nachbedingung von O,
und es gilt $M =^{?} M'$.

(3) O ist ein Nutzer von M
:<=>
M' ist eine Vorbedingung von O,
und es gilt $M' =^{!} M$.

(4) O ist ein Zerstörer von M
:<=>
M' bildet den nichtnegierten Part einer löschenden,
d.h. negierten Nachbedingung von O,
und es gilt $M' =^{?} M$.

Aussage des Erfüllungskriteriums (mit Bezug auf Abb. 3.3-1):

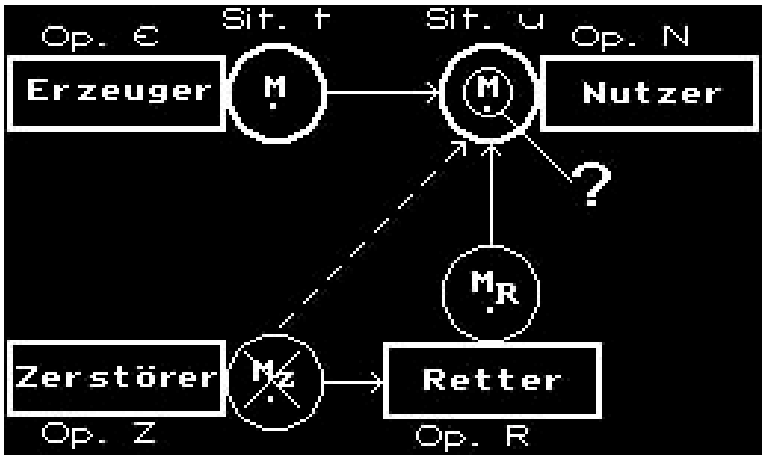


Abb. 3.3-1 Veranschaulichung des Erfüllungskriteriums

Ein Merkmal M ist notwendigerweise in einer Situation u erfüllt.
<=>
- In einer Situation t, welche entweder mit u identisch ist oder eine Vorgängersituation zu u darstellt, ist M notwendigerweise erfüllt,

und
- für jede Operator-Instanz Z, durch deren Anwendung ein Merkmal M_Z gelöscht wird, wobei $M =^{?} M_Z$, gilt:
  * Z wird schon vor der erzeugenden Operator-Instanz E (d.h. $Z <_n E$) angewandt, oder
  * Z wird erst nach der nutzenden Operator-Instanz N (d.h. $N <_n Z$) angewandt, oder
  * es existiert eine als "Retter" fungierende Operator-Instanz R, welche

Nachfolger von Z (d.h. $Z <_n R$) und Vorgänger von N (d.h. $R <_n N$) ist, also bereits vor Erreichen von u angewandt wird, und durch deren Ausführung ein Merkmal M_R erzeugt wird, wobei folgende Implikation gilt:
$M =^! M_Z \Rightarrow M =^! M_R$.

Anm.: - Situation t stellt die Situation dar, die sich unmittelbar nach Anwendung von Operator-Instanz E ergibt.
Situation u stellt die Situation dar, die unmittelbar vor Anwendung von Operator-Instanz N erreicht sein muß.
- Durch die vorausgegangenen Definitionen von Vorgänger (siehe Def. 3.1-1), Codesignation bzw. Codesigniertheit (siehe Def. 3.2-1 und Def. 3.2-2) und Codesignierbarkeit (siehe Def. 3.3-1 und Def. 3.3-2) kann bei der Formulierung des Erfüllungskriteriums im Vergleich zu [Chapman 1987] der etwas verwirrende Gebrauch der Adverbien 'notwendigerweise' und 'möglicherweise', den dieser Planungsansatz notwendig macht, auf die Gültigkeit von Merkmalen beschränkt bleiben.

Auf Grund der gewählten einfachen Aktionsrepräsentation läßt sich an Hand des obigen Kriteriums die Gültigkeit eines Merkmals an einer bestimmten Stelle im Plan eindeutig ermitteln, ohne eventuelle zusätzliche Seiteneffekte berücksichtigen zu müssen. Die Voraussetzung für die Vollständigkeit und Korrektheit bei der Plangenerierung ist dadurch gegeben.

3.3.2 Erreichen von (Teil-)Zielen

Die Idee beim nichtlinearen Planen, die auch von TWEAK aufgegriffen wird, besteht, wie bereits gesagt, in einer unabhängigen Betrachtung von an einer bestimmten Stelle im Plan zu erfüllenden Teilzielen.

Die Vorgehensweise bei der Bearbeitung eines solchen Teilzieles M, etwa eines der im Rahmen der Problemstellung spezifizierten Zielmerkmale, basiert dann auf dem Erfüllungskriterium und läßt sich in verschiedene Phasen oder Schritte untergliedern:

Schritt 1: Zunächst muß die Gültigkeit des betrachteten Merkmals analysiert werden.

Ist M bereits notwendigerweise erfüllt, so ist keine weitere Planarbeit erforderlich und das nächste Teilziel kann ausgewählt und bearbeitet werden.

Schritt 2: Andernfalls muß nach einem erzeugenden Operator gesucht werden. Hierbei gibt es zwei Möglichkeiten:

(1) Es wird im bestehenden Plan, falls vorhanden, ein bereits eingebauter Operator, mit anderen Worten eine Operator-Instanz, gewählt, die das betrachtete Merkmal zu erfüllen in der Lage ist, d.h. die ein potentieller Erzeuger von M darstellt und zu deren Vorgänger-Instanzen nicht die Operator-Instanz zählt, deren Vorbedingung u.a. das betrachtete Merkmal M ist.

(2) Es wird, falls vorhanden, eine der verfügbaren anwendbaren Operator-Schablonen in den aktuellen Plan eingebaut, die wiederum natürlich ein potentieller Erzeuger sein muß. Der Plan wird also um eine Operator-Instanz erweitert.

Es scheint naheliegend, die Strategie bei der Plangenerierung so zu gestalten, daß es vermieden wird, neue Operatoren in den bestehenden Plan einzubauen, falls erzeugende Operator-Instanzen bereits existieren, da mit jedem neuen

Planschritt, der auch auf andere Ziele als Zerstörer wirken kann, der resultierende Plan "aufgeblähter" wird. Darüber hinaus müssen die neu hinzugekommenen Vorbedingungen ihrerseits erfüllt werden, was zusätzliche Planarbeit mit sich bringt.

Die Optimalität des jeweils generierten Lösungs-Plans - auch TWEAK erhebt den Anspruch, ein optimal arbeitender Planer zu sein - ist aber nur dann gewährleistet, wenn beide Möglichkeiten betrachtet werden: Der zunächst unnötig erscheinende Einbau eines neuen Operators kann sich im weiteren Planungsverlauf als vorteilhafter oder sogar als notwendig erweisen.

Bei der Lösung des Sussman-Problems mit den Operatoren PUTON und NEWTOWER (siehe Abschnitt 7.1) ergibt sich ein Zwischenstadium im Laufe des Planungsprozesses, bei dem zur Erreichung des optimalen Lösungs-Plans (siehe 7.1.3) ein neuer Operator (NEWTOWER) in den betrachteten unvollständigen nichtlinearen Plan eingebaut werden muß, obwohl eine in diesem "Plan-Gerüst" bereits bestehende Operator-Instanz (PUTON) als Erzeuger hätte fungieren können.

Schritt 3: Sind mögliche alte und neue Erzeuger bestimmt, so müssen noch entsprechende Planmodifikationen bezüglich Variablenbindung und Operator-Ordnung durchgeführt werden, um den jeweils betrachteten Erzeuger zu etablieren und die Gültigkeit des betrachteten Teilziels notwendig werden zu lassen.Dies wird durch das Setzen entsprechender Wert- und Zeit-Constraints bewerkstelligt.

Es ist zu betonen, daß für jeden bestimmten Erzeuger ein neuer Plan generiert wird, welcher sich von dem ursprünglich betrachteten Plan mit dem unerfüllten Teilziel M darin unterscheidet, daß nun für M ein Erzeuger eingetragen ist. Die resultierenden Pläne werden in einer Warteschlange gemäß der Breitensuchstrategie verwaltet und in nachfolgenden Schritten unabhängig voneinander nach weiteren unerfüllten Planzielen betrachtet.

Die bisher behandelten Schritte bei der Bearbeitung von Teilzielen seien durch das folgende Beispiel (Abb. 3.3-2 bis 3.3-4) verdeutlicht:

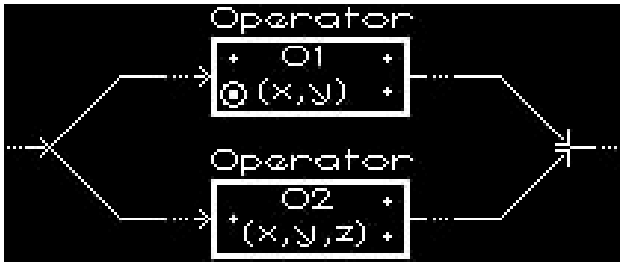


Abb. 3.3-2 Beispiel eines Plan-Ausschnitts

Gegeben sei obiger Ausschnitt (Abb. 3.3-2) eines unvollständigen nichtlinearen Plans: Die Operator-Instanz O1 sei ungelöst, da eine ihrer Vorbedingungen (im Bild durch Umrandung markiert) nicht mehr auf Grund irgendwelcher Interaktionen mit anderen Teilplänen oder noch nicht wegen fehlender vorausgehender Erzeugung erfüllt ist (-> Schritt 1). Diese Vorbedingung von O1 (=: V2) stellt also das aktuell betrachtete Teilziel dar.

Die Suche nach Erzeugern (-> Schritt 2) liefere folgendes Ergebnis: Operator-Instanz O2 weist eine nichtnegierte Nachbedingung, einen erzeugenden Effekt auf,

welcher mit V2 codesignierbar ist, daher handelt es sich bei O2 um einen potentiellen alten, da bereits eingebauten Erzeuger. Weitere erzeugende Operator-Instanzen seien nicht im Plan vorhanden. Darüber hinaus sei eine der verfügbaren Operator-Schablonen anwendbar mit demselben Effekt.

Schließlich müssen die ermittelten Erzeuger als solche in den Plan eingetragen werden (-> Schritt 3), d.h. Linearisierung, Wert-Bindung und Operator-Einbau sind durchzuführen. Die Abbildungen 3.3-3 und 3.3-4 zeigen die resultierenden nichtlinearen Pläne.

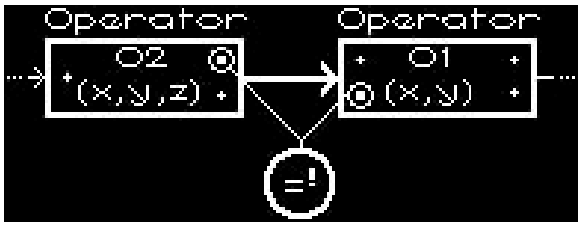


Abb. 3.3-3 O2 als Erzeuger des betrachteten Teilziels V2, einer Vorbedingung von O1

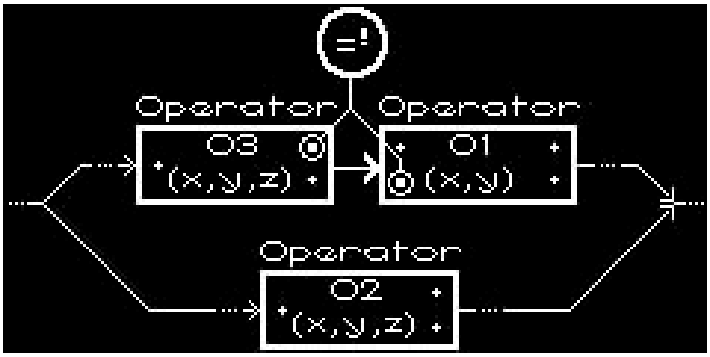


Abb. 3.3-4 Die neu eingebaute Operator-Instanz O3 als Erzeuger für das betrachtete Teilziel

Die eingeführten Wert-Constraints sind bei den beiden obigen Darstellungen nicht wie in vorausgegangenen Beispielen zwischen Argumenten gesetzt, sondern zwischen Merkmalen. Dies entspricht der tatsächlichen Vorgehensweise des Planers; die Argument-Wert-Bindungen ergeben sich als unmittelbare Konsequenz daraus.

Wenn z.B. die beiden Merkmale (steht_auf x y) und (steht_auf y z) zweier Operator-Instanzen O1 und O2 per Codesignation-Wert-Constraint miteinander verbunden werden, dann wirkt sich dies als Folge der Unifikation so aus, daß x[O1] =! y[O2] und y[O1] =! z[O2] gilt, d.h. die jeweils korrespondierenden Merkmal-Argumente codesignieren:

```
        M E R K M A L  1              (steht_auf  x y)
               |                                 | |
Codesignation-Wert-Constraint   ==>              = =
               |                                 | |
        M E R K M A L  2              (steht_auf  y z)
```

Noch sind nicht alle Maßnahmen durchgeführt, um das betrachtete Teilziel abhaken und zur Bearbeitung neuer noch unerfüllter Merkmale übergehen zu können. Es fehlen noch die Schritte Konflikt-Erkennung und -Auflösung.

Schritt 4: Es genügt nicht, nur allein für die Erzeugung zu sorgen, um die Gültigkeit eines betrachteten Merkmals an einer bestimmten Stelle im

Plan zu gewährleisten, sondern mögliche störende Interaktionen müssen als solche erkannt und dann in einem nachfolgendem Schritt aufgelöst werden.

Sämtliche Operator-Instanzen im Plan müssen daraufhin untersucht werden, ob sie als Zerstörer bezüglich des gerade betrachteten Teilziels wirken können.

Beispielsweise könnte in obigem Planausschnitt (Abb. 3.3-4) die Operator-Instanz O2 auf Grund der bestehenden Ordnung als Zerstörer auf die betrachtete Vorbedingung von O1 wirken, wenn O2 über eine negierte, d.h. löschende Nachbedingung verfügt, deren nichtnegierter Part mit dieser Vorbedingung codesignierbar ist.

Bei Einbau eines neuen Operators zur Erzeugung eines betrachteten Teilziels muß außerdem untersucht werden, ob eben dieser auf andere bereits erfüllte Teilziele störend wirken könnte. Auch in solchen Fällen ist eine entsprechende Auflösung nötig.

Schritt 5: Es gilt nun noch, die aufgetretenen und als solche erkannten Konflikte durch geeignete Methoden aufzulösen. Um der Schwierigkeit dieser Aufgabe Ausdruck zu verleihen, sei diesem planungsspezifischen Problem ein eigener nun folgender Unterabschnitt gewidmet.

3.3.3 Methoden zur Konflikt-Auflösung

Die Vorgehensweise von TWEAK bei auftretenden Konflikten basiert ebenfalls auf dem Erfüllungskriterium.

Eine Konflikt-Situation (kurz: ein Konflikt) liegt allgemein dann vor, wenn ein zur Erfüllung eines Teilziels eingebauter Operator u.a. auch einen löschenden Effekt aufweist, welcher mit einem an einer anderen Stelle im Plan zu erfüllenden Teilziel codesignierbar ist, und die bestehende (partielle) Operator-Ordnung dem Konflikt nicht entgegenwirkt.

Daß solche Teilzielinteraktionen (subgoal interactions) im allgemeinen unvermeidbar sind, liegt an der Tatsache, daß die Linearitätsannahme, wie bereits im zweiten Kapitel hervorgehoben, eigentlich nicht richtig ist und ein zusammengesetztes Planziel im allgemeinen nicht einfach dadurch gelöst werden kann, daß es in seine Teilziele zerlegt wird und deren Lösungen anschließend zusammengesetzt werden, sondern zusätzliche Planarbeit ist häufig notwendig, da die Teillösungen sich gegenseitig beeinflussen können.

Als mögliche Konfliktlösungs-Methoden lassen sich die folgenden unterscheiden, die dann im Anschluß näher erläutert und an Hand eines Beispiels veranschaulicht werden:

(1) Vorverlegen (demotion),
(2) Nachverlegen (promotion),
(3) Separation (separation),
(4) Retter einschieben (white knight).

Zur Erläuterung der Vorgehensweise bei den verschiedenen Lösungsansätzen diene die in Abb. 3.3-5 skizzierte Konflikt-Situation bzw. deren Lösungsvarianten in Abb. 3.3-6 bis 3.3-10:

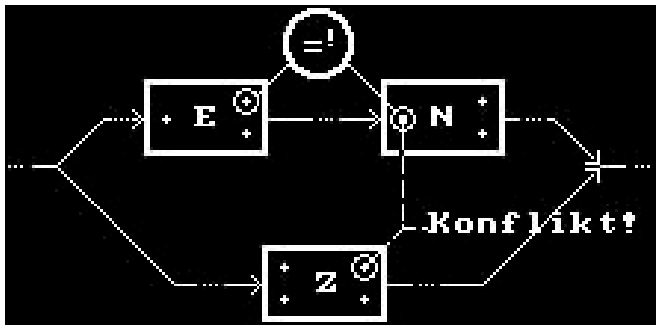


Abb. 3.3-5 Beispiel einer Konflikt-Situation:
E steht für Erzeuger,
N für Nutzer und
Z für Zerstörer

Zu (1): Ein als Zerstörer wirkender Planschritt kann dadurch neutralisiert werden, daß er vor die erzeugende Instanz im Plan verlegt wird, so daß die Anwendung desselben keine Auswirkung mehr auf die Erzeuger-Nutzer-Abhängigkeit hat (Abb. 3.3-6).

Diese Planmodifikation in Form einer Linearisierungsmaßnahme ist nur möglich, falls sie zur bestehenden (partiellen) Operator-Ordnung verträglich ist. Eine derartige Ordnungs-Verschärfung, um einer Interaktion entgegenzuwirken, kann naheliegenderweise als Vorverlegen (demotion) bezeichnet werden.

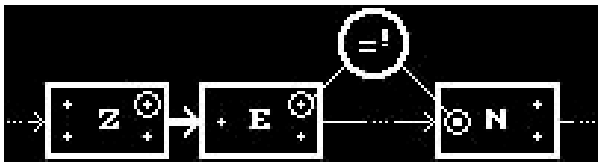


Abb. 3.3-6 Konflikt-Auflösung durch Vorverlegen des Zerstörers

Zu (2): Analog zu (1) kann zur Konflikt-Auflösung auch derart linearisiert werden, daß der Zerstörer erst nach der nutzenden Instanz angewandt wird (Abb. 3.3-7). Dies ist wiederum nur möglich, wenn die bestehende Operator-Ordnung dies zuläßt. Man spricht hierbei von Nachverlegen (promotion).

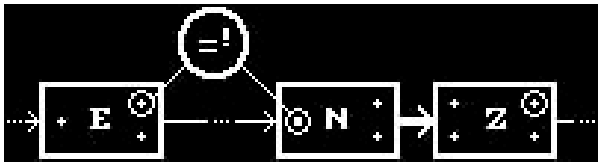


Abb. 3.3-7 Konflikt-Auflösung durch Nachverlegen des Zerstörers

Zu (3): Durch das Setzen eines entsprechenden Wert-Constraints (Abb. 3.3-8), genauer: Noncodesignation-Wert-Constraints, kann auch die Unifizierbarkeit bzw. Codesignierbarkeit zwischen löschendem Zerstörer-Effekt und betrachtetem Teilziel explizit durch Separation verboten werden, falls dies zu den bestehenden Wert-Bindungen der im Plan enthaltenen Variablen konsistent möglich ist.

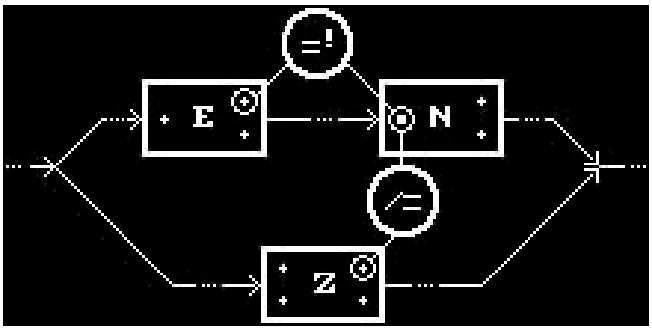


Abb. 3.3-8 Konflikt-Auflösung durch Separation

Zur Erinnerung sei hier noch einmal beispielhaft angeführt, welche Wirkung ein Noncodesignation-Wert-Constraint zwischen zwei Merkmalen, etwa der Form (steht_auf x y) und (steht_auf y z), zweier Operator-Instanzen O1 und O2 hat: Das gesetzte Wert-Constraint wirkt sich als Folge des "Unifikations-Verbots" so aus, daß x[O1] $/=^{!}$ y[O2] oder y[O1] $/=^{!}$ z[O2] gilt, d.h. mindestens ein Paar korrespondierender Merkmal-Argumente noncodesigniert.

Zwischen zwei Merkmalen kann ein Noncodesignation-Wert-Constraint nur dann gesetzt werden, wenn die beiden Merkmale noncodesignierbar sind.

Weiteres Bsp.: Für ein Merkmalpaar (steht_auf A B) und (steht_auf A x), wobei A und B Konstanten sind und x eine Variable darstellt, die mit B codesigniere, gilt, daß es nicht noncodesignierbar ist, da es kein Paar korrespondierender Merkmal-Argumente gibt, die noncodesignierbar sind, denn A $=^{!}$ A und x$=^{!}$ B, hier im Beispiel.

Es handelt sich also um eine disjunktivische Forderung, während ein gesetztes Codesignation-Wert-Constraint zwischen zwei Merkmalen eine konjunktivische Forderung darstellt.

Zu (4): Schließlich kann auch immer eine als "Retter" (white knight) fungierende Operator-Instanz zwischen Zerstörer und nutzende Instanz eingeschoben werden.

Der Vorteil dieser zunächst umständlich, da u.U. Plan vergrößernd, wirkenden Methode des Retter Einschiebens besteht darin, daß dazu immer die Möglichkeit gegeben ist, während Verlegung des Zerstörers oder Separation in bestimmten Fällen nicht durchgeführt werden kann.

Es lassen sich zwei verschiedene Varianten dieser Konfliktlösungs-Methode unterscheiden:

(1) "Alten" Retter einschieben:
Ein im Plan bereits eingebauter Operator wird, falls existent, als Retter bestimmt. Diese Variante ist allerdings nur dann gegeben, wenn die bestehende partielle Operator-Ordnung so verschärft werden kann, daß die Position des Retters im Plan zwischen Zerstörer und Nutzer zu liegen kommt.

(2) "Neuen" Retter einschieben:
Eine der verfügbaren anwendbaren Operator-Schablonen wird zur Konflikt-Auflösung in den Plan eingebaut und somit zur neuen Operator-Instanz, die mittels Zeit-Constraints im Plan notwendigerweise zwischen Zerstörer und Nutzer positioniert wird.

Letztere Methode führt immer zum gewünschten Ziel, nämlich zur Behebung der störenden Interaktion, und es gibt auch keinerlei Einschränkungen bezüglich der Voraussetzungen, die erfüllt sein müssen, damit ein neuer Retter eingeschoben werden kann. Allerdings ist diese Strategie nicht gerade ein Garant für die Optimalität des generierten Lösungs-Plans, da der eingebaute Retter zum einen neue Konflikte verursachen kann und zum andern zusätzliche zu erfüllende Teilziele mit sich bringt. Häufig führen andere Ansätze schneller zur gesuchten Lösung und resultieren auch in einer kompakteren Form der Problemlösung.

Ist ein Retter bestimmt, so müssen neben der bereits angesprochenen Ordnungs-Verschärfung gegebenenfalls auch Wertebereichs-Einschränkungen der beteiligten Variablen festgelegt werden, wobei die Art bzw. das Vorhandensein dieser Wert-Bindungen mittels Codesignation-Wert-Constraint davon abhängt, ob der Zerstörer-Effekt mit dem betrachteten Teilziel (tatsächlich) codesigniert oder ob nur eine potentielle Codesignierbarkeit ohne gleichzeitiger Codesigniertheit zwischen beiden Merkmalen vorliegt.

Dies entspricht dem Grundsatz des Constraint-Posting-Ansatzes, nämlich so wenig Festlegungen zu treffen, wie nötig.

Der Retter muß nur dann mit dem Nutzer "in Verbindung treten", d.h. Retter-Effekt und betrachtetes Teilziel müssen nur dann mit Hilfe eines Codesignation-Wert-Constraints zueinander in Wert-Relation gebracht werden, wenn dies auf Grund der Codesigniertheit zwischen Zerstörer-Effekt und Teilziel erforderlich wird, damit die Gültigkeit des Vorbedingungs-Merkmals notwendigerweise trotz Zerstörers erhalten bleibt.

Sind Zerstörer-Effekt und Teilziel aber nur codesignierbar und nicht codesigniert, so bedarf es zunächst keiner entsprechenden Wert-Bindung, sondern zu dieser Maßnahme muß, falls überhaupt, erst im weiteren Planungsverlauf gegriffen werden, wenn dann doch Zerstörer-Effekt und betrachtetes Teilziel codesignieren.

Dieser Sachverhalt wird in den Abb. 3.3-9 und 3.3-10 darzustellen versucht.

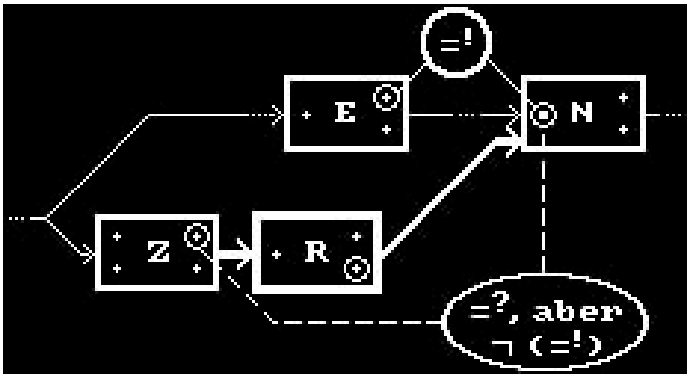


Abb. 3.3-9 Konflikt-Auflösung durch Einschieben eines Retters:

(Noch) keine Constraint-Bindung des erzeugenden Retter-Effekts zum betrachteten Teilziel, der Vorbedingung von N, da Zerstörer-Effekt und betrachtetes Teilziel codesignierbar, aber nicht codesigniert sind.

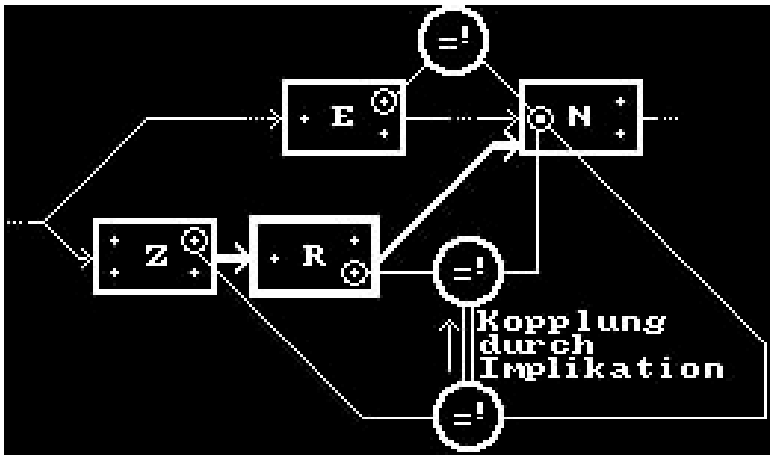


Abb. 3.3-10 Konflikt-Auflösung durch Einschieben eines Retters:

Retter-Effekt und betrachtetes Teilziel stehen nun explizit durch Wert-Constraint in Codesignations-Relation zueinander, da auch Zerstörer-Effekt und Teilziel (nun) codesignieren.

TWEAKs Planungsstrategie sieht so aus, daß ein betrachtetes Teilziel an einer bestimmten Stelle im Plan dann als erfüllt gilt, wenn seine Gültigkeit an dieser Stelle notwendigerweise gewährleistet ist. Um dies nun auch bei auftretenden Konflikten garantieren zu können, die durch einen zwischengeschobenen Retter aufgelöst werden, bedarf es der expliziten Repräsentation im Plan folgender Implikation via Constraints:

(Die gewählten Bezeichnungen entsprechen denen, die bei der Darstellung des Erfüllungskriteriums auf S. 27 verwendet wurden:
M bezeichne das betrachtete Teilziel, M_Z den löschenden Zerstörer-Effekt und M_R den erzeugenden Retter-Effekt):

$$M =^{!} M_Z \Rightarrow M =^{!} M_R.$$

Eine laxe Formulierung dieser Implikation könnte folgendermaßen lauten: Der Retter wird erst dann "aktiviert", wenn Zerstörer-Effekt und Teilziel codesignieren, also tatsächlich ein Konflikt vorliegt.

Naheliegenderweise bietet sich auch die Vorgehensweise an, den eingeschobenen Retter auf jeden Fall zu aktivieren, also unabhängig von $M =^{?} M_Z$ oder $M =^{!} M_Z$ via Codesignation-Wert-Constraint $M =^{!} M_R$ zu setzen. Dadurch wäre die störende Interaktion auf jeden Fall behoben. Es würde aber auch in solchen Fällen $M =^{!} M_R$ gesetzt werden, wenn es eigentlich (noch) nicht erforderlich wäre.

Entsprechend der Leitidee des Constraint-Posting-Ansatzes, die auch hinter der Formulierung des Erfüllungskriteriums steht, wird auch in diesem Fall so wenig wie möglich festgelegt; die schwächste Forderung wird gestellt, die gerade noch zum erwünschten Ziel führt.

Durch die strikte Einhaltung des Treffens von Entscheidungen nur gemäß Erfüllungskriterium wird die Vollständigkeit bei der Plangenerieung durch TWEAK garantiert. Die Darstellung dazu notwendiger Implikations-Constraints erfordert zwar zusätzlichen Aufwand, bereitet aber keine grundsätzlichen Probleme.

Nachdem nun sämtliche mögliche Konfliktlösungs-Methoden vorgestellt wurden, stellt sich die Frage, welche Methode in welcher Konflikt-Situation angebracht erscheint und zum bestmöglichen Resultat führt. Leider läßt sich nur sagen, daß meist verschiedene Lösungsansätze möglich sind und vorab keine Aussage über die "optimale Konflikt-Auflösung" getroffen werden kann. Es gibt nur bestimmte Situationen, in denen manche Methoden auf Grund der bestehenden (partiellen) Operator-Ordnung oder der Wert-Bindungen der im Plan enthaltenen Variablen nicht angewandt werden können. Wie solche Situationen gartet sein müssen, wird in [Chapman 1987] nicht näher behandelt. Daher möchte ich Erläuterungen hierzu auf die Beschreibung des im Rahmen dieser Diplomarbeit implementierten Planers (siehe Kap. 4; genauer: Unterabschnitt 4.3.4) verlagern.

Die Wahl der jeweils am besten geeigneten Konfliktlösungs-Methode trifft TWEAK jeweils indeterministisch (siehe dazu auch Abschnitt 3.4).

3.3.4 Zielauswahl auf übergeordneter Kontrollebene

Den Anstoß zur Planung gibt stets ein zu erfüllendes Teilziel, das auf einer übergeordneten Kontrollebene im jeweiligen Planungsstadium bestimmt wird. Sind durch den generierten Plan sämtliche Ziele erfüllt, so ist das Problem gelöst, und die Planung terminiert.

Zu Beginn des Planungsprozesses bilden die durch die Zielsituation spezifizierten Merkmale die ersten zu erfüllenden Teilziele. Zu ihrer Lösung werden Operatoren in den bestehenden Plan eingebaut (Vorgehen gemäß Rückwärtssuche und Mittel-Ziel-Analyse), die ihrerseits wieder weitere Planarbeit mit sich bringen, da mit jeder neuen Operator-Instanz neue Teilziele zu erfüllen sind, die die Vorbedingungen der jeweiligen Instanz darstellen.

Im allgemeinen sind an einer Stelle im Plan mehrere Teilziele gleichzeitig zu bearbeiten. Darüber hinaus gibt es meist nicht nur eine derartige Stelle, im folgenden als ungelöster (Plan-)Knoten bezeichnet, sondern mehrere Operator-Instanzen (oder Planknoten) weisen häufig in einem bestimmten Planungsstadium unerfüllte (Vorbedingungs-)Merkmale auf. Die Gleichzeitigkeit der Bearbeitung verschiedener Teilziele wird durch den nichtlinearen Planungsansatz realisiert. Dennoch erfolgt während der Planung zunächst eine sequentialisierte Betrachtung der einzelnen Teilziele, die dann schließlich in einem nichtlinearen Teilplan resultiert.

Die Wahl des jeweils als nächstes zu betrachtenden Planknotens wird in gewissem Maße durch die Strategie der Rückwärtssuche vorgegeben. Ausgehend vom Zielknoten werden generationsweise Operator-Instanzen auf unerfüllte Merkmale hin untersucht. Dabei bilden Knoten mit gleichem Abstand, definiert über die Vorgänger-Relation, zum Zielknoten eine Generation. In der nachfolgenden Abbildung 3.3-11 bilden etwa A und B (mit Abstand 1 zum Zielknoten) oder auch C, D und E (mit Abstand 2 zum Zielknoten) je eine Generation von Planknoten. Der als nächster betrachtete Planknoten wird aus der Knoten-Generation gewählt, die unter den Generationen mit ungelösten Knoten den geringsten Abstand zum Zielknoten aufweist. Im gegebenen Beispiel (Abb. 3.3-11) wäre dies die aus den ungelösten Planknoten A und B bestehende Generation 1.

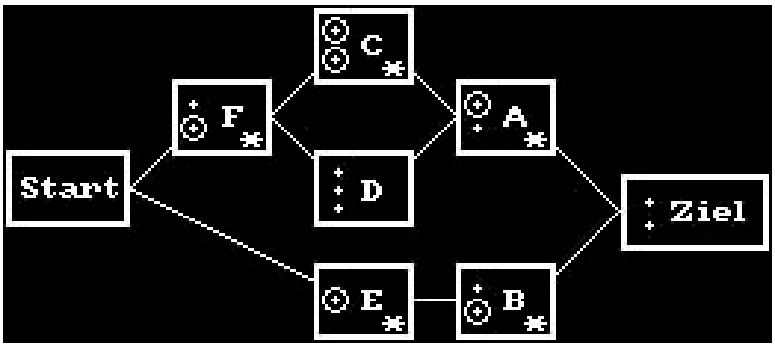


Abb. 3.3-11 Beispiel eines unvollständigen nichtlinearen Plans

(Die Knoten symbolisieren Operator-Instanzen, die Kanten repräsentieren die Operator-Ordnung; ungelöste Knoten sind mit '*' gekennzeichnet, unerfüllte (Vorbedingungs-)Merkmale durch Umrandung markiert.)

Das dargestellte Beispiel (Abb. 3.3-11) macht auch deutlich, daß die Zugehörigkeit eines Knotens nicht auf eine Generation beschränkt zu sein braucht: Der Start-Knoten zählt sowohl zur Generation 3 (zusammen mit Knoten F) als auch zur Generation 4. Dieser Sachverhalt kann aus Effizienzgründen bei der Implementierung ausgenützt werden, hat aber ansonsten keine größere Bedeutung.

Die Bestimmung der zu betrachtenden Generation mit noch ungelösten Knoten gestaltet sich also durch die Rückwärts-Suchstrategie deterministisch. Hingegegen bleibt die Wahl des ungelösten Knotens innerhalb dieser bestimmten Generation offen: Auf Grund des Constraint-Posting-Ansatzes und der gewählten elementaren Aktions-Repräsentation (siehe dazu auch Abschnitt 3.5) kann TWEAK diese Wahl indeterministisch treffen. Im obigen Beispiel würde also A oder B als nächste zu betrachtende ungelöste Operator-Instanz gewählt werden. Die Wahl hat keine Auswirkung auf das Ergebnis, die Menge ungelöster Knoten innerhalb eines Plans wird via Breitensuche vollständig durchsucht.

Ein Knoten wird dann als ungelöst bezeichnet, wenn er mindestens ein unerfülltes Merkmal aufweist. Bei der Wahl von A im gegebenen Beispiel (Abb. 3.3-11) würde die Bestimmung des als nächstes zu betrachtenden Teilzieles keine Probleme bereiten, da nur ein unerfülltes Vorbedingungs-Merkmal vorliegt. Häufig liegen allerdings mehrere noch nicht erfüllte Vorbedingungen vor, beispielsweise beim Planknoten C in Abb. 3.3-11. Auch diese Wahl kann bei dem von TWEAK verfolgten Planungsansatz indeterministisch erfolgen, gestaltet sich also gleichermaßen wie die Wahl des nächsten zu lösenden Planknotens.

3.4 TWEAKs Indeterminismus beim Lösen von Problemen

Beim Lesen der Beschreibung von TWEAK [Chapman 1987] fällt auf, daß immer wieder die indeterministische Vorgehensweise dieses Planers betont wird. Daher soll auch diesem Punkt ein eigener, wenn auch kurzer Abschnitt gewidmet sein, um die Voraussetzungen und Konsequenzen darzulegen, die sich aus diesem Ansatz ergeben.

Die Strategie bei der Plangenerierung durch TWEAK läßt sich dadurch charakterisieren, daß immer dann, wenn eine Auswahl getroffen werden muß, kein

heuristischer Weg eingeschlagen wird, der schnell zum Erfolg führen, aber auch in einer Sackgasse enden kann, sondern in Frage kommende Alternativen zunächst gleichberechtigt behandelt werden, um nicht eine eventuell optimale Lösung durch eine frühzeitige Fehlentscheidung unter den Tisch fallen zu lassen oder gar die einzige Lösung eines gestellten Problems überhaupt nicht zu finden und das Probelm fälschlicherweise als unlösbar zu deklarieren. Die Konsequenzen, die sich daraus ergeben, lassen sich kurz durch Attribute wie Optimalität und Vollständigkeit, aber auch Ineffizienz umreissen.

Zum besseren Überblick kann zwischen verschiedenen Indeterminismen differenziert werden, nachdem in Abschnitt 3.3 eine detaillierte Beschreibung der einzelnen Planungstätigkeiten vorausging. Eine indeterministische Auswahl erfolgt bei der Zielauswahl auf übergeordneter Kontrollebene, bei der Bestimmung möglicher Erzeuger für das jeweils gewählte unerfüllte Merkmal und bei der Wahl möglicher Konfliktlösungs-Methoden zur Behebung störender Interaktionen. Es ist an dieser Stelle noch anzumerken, daß diese einzelnen Vorgänge ineinander übergreifen und somit ein "verschachtelter Indeterminismus" vorliegt, der im wesentlichen durch drei Hilfsmittel auf einigermaßen praktikable Weise handhabbar gemacht wird:

- Breitensuch-Strategie auf der obersten Kontrollebene des Planers,
- Einbetten der Operator-Instanzen in ein nichtlineares Ordnungsgefüge,
- Constraint-Posting-Ansatz, wodurch mehrere Möglichkeiten gleichzeitig, d.h. durch einen unvollständigen Plan darstellbar werden.

Die Hauptschwierigkeit ergibt sich aus der Größe des Suchraums möglicher Pläne, welcher als Folge der indeterministischen Vorgehensweise exponentiell anwächst. Die Breitensuch-Strategie auf oberster Kontrollebene von TWEAK ist zwar ein Garant für Optimalität und Vollständigkeit, läßt aber die Rechenzeiten für die Suche explodieren.

Es scheint naheliegend, gewisse Heuristiken einbringen zu können, ohne von der Zielsetzung wegzukommen, einen Planer zu erhalten, der sich durch die angestrebten Attribute auszeichnet. Die dadurch gewonnene Effizienzsteigerung könnte realistischere Einsatzmöglichkeiten eröffnen, als dies mit TWEAK der Fall ist. Es müßte sich dabei aber um möglichst nicht bereichsspezifische Heuristiken handeln, um die Domänen-Unabhängigkeit zu bewahren. Die Probleme, die sich bei der Formulierung solcher Heuristiken und den sich daraus resultierenden Konsequenzen ergeben, seien an Hand eines Beispiels verdeutlicht:

Beispiel:

Heuristik 1: Wenn alte und neue Erzeuger zur Erfüllung eines Teilziels anwendbar sind, dann wähle nur die alten.

Heuristik 2: Wenn bei auftretenden störenden Interaktionen die Voraussetzungen auch für andere Konfliktlösungs-Methoden als 'Retter einschieben' gegeben sind, dann vermeide letztere Lösungsvariante.

Die Intention, die hinter diesen beiden Heuristiken stehen könnte, ist die Generierung von bezüglich der Anzahl eingebauter Operatoren möglichst optimalen Plänen. Wenn irgendwie möglich, soll vermieden werden, daß neue Operator-Instanzen hinzukommen, da dies wieder zusätzliche Planarbeit mit sich bringen und die Optimalität des Lösungsplans gefährden könnte; falls durch Linearisierungsmaßnahmen und/oder Wert-Bindungen zunächst gleiche Effekte erzielbar sind, so ist diesen der Vorzug zu geben und die anderen Varianten, die mit dem Einbau weiterer Planknoten verbunden wären, sind nicht in Betracht zu ziehen.

Diese naheliegende Vorgehensweise birgt aber die Gefahr, daß dadurch manche Möglichkeiten während der Plangenerierung unbetrachtet bleiben, die zunächst auf Grund der scheinbar unnötigen Einführung einer neuen Operator-Instanz mehr Aufwand kosten als alternative Möglichkeiten, sich aber vielleicht im weiteren Planungsverlauf als zur optimalen Lösung führend herausstellen.

Bei der Lösung des Sussman-Problems (siehe Abschnitt 6.1) kommen beide angesprochenen Fälle vor: In einer Situation stellt es sich im nachhinein als notwendig heraus, um die optimale Lösung zu erhalten, einen neuen Erzeuger zu wählen, obwohl im bestehenden Plan bereits einer verfügbar gewesen wäre; in einer anderen Situation wird es nötig, zur Auflösung eines erkannten Konfliktes einen neuen Retter einzuschieben, obgleich Separation ebenso und mit zunächst gleicher Wirkung durchführbar gewesen wäre.

TWEAK erweist sich als nicht effizient genug, um komplexere Probleme lösen zu können. Aber dies lag auch nicht in der Absicht seines Erstellers. Die Zielsetzung lag darin, die dem Planen inhärente Problematik des Indeterminismus aufzugreifen und ein Planungssystem zu implementieren, das vollständig den Raum möglicher Lösungspläne durchsucht. Das Aufwandsproblem der kombinatorischen Explosion stellt dabei eine praktisch zwar unübersehbare, aber vor diesem theoretischen Hintergrund doch sekundäre Hürde dar.

3.5 Restriktive Aktionsrepräsentation in TWEAK

Wenn auch die Domänen-Unabhängigkeit verwirklicht ist, so sind dennoch die Repräsentationsmöglichkeiten TWEAKs zu wenig ausdrucksstark, so daß realistischere Problemstellungen meist nicht in einer für den Planer verständlichen Form dargestellt werden können.

Die häufige Disjunktheit innerer Eigenschaften von Repräsentationsformalismen wie z.B Vollständigkeit, Korrektheit, Effizienz, Mächtigkeit (Ausdrucksstärke) ist ein bekanntes Problem: die relativ ausdrucksstarke Prädikatenlogik zweiter Stufe ist weder vollständig noch entscheidbar, die weniger ausdrucksstarke Prädikatenlogik erster Stufe ist zwar auch nicht entscheidbar, aber immerhin vollständig, und die einfache Aussagenlogik weist sowohl die Vorzüge der Vollständigkeit, als auch die der Entscheidbarkeit auf, nur ist hierbei der Abstraktionsgrad zu hoch, als daß damit Sachverhalte komplexerer Domänen darstellbar wären.

Bei der Wahl einer geeigneten Aktionsrepräsentation gilt es also, einen geeigneten Kompromiß zu finden. Bei einem Planer für eine spezielle Domäne kann der gewählte Formalismus auf den jeweiligen Problembereich abgestimmt sein. Eine entscheidende Rolle spielt dabei, ob ein Planer nur spezielle Probleme einer ausgewählten Domäne lösen soll, dabei aber auf Effizienz geachtet wird, oder ob ein Planer dazu in der Lage sein soll, ein breiteres Spektrum von Problemstellungen bearbeiten zu können. Bei TWEAK fiel diese Entscheidung zugunsten der Domänen-Unabhängigkeit und Vollständigkeit, sehr zum Nachteil von Mächtigkeit und Effizienz bezüglich Laufzeitverhalten und Speicherbedarf.

Die Aussagekraft des Erfüllungskriteriums, auf das sich die gesamte Vorgehensweise bei der Generierung von Plänen durch TWEAK stützt, basiert auf diesem gewählten Formalismus. Die in einer bestimmten Situation gültigen Merkmale werden nicht als solche explizit verwaltet, sondern jeweils mit Hilfe des Erfüllungskriteriums mit polynomiellem Aufwand ermittelt.

Es muß gewährleistet sein, daß sich keine Nebeneffekte einschleichen können, denn dadurch entstünden nicht berechenbare Situationen. Dies erfordert, daß sämtliche Veränderungen der Welt, hervorgerufen durch Anwendung von Operatoren, explizit in den Nachbedingungen der angewandten Operatoren auftauchen müssen. Darüber hinaus müssen die Merkmale generell atomaren Aufbau haben. Funktionen

oder Quantoren in den Vor- oder Nachbedingungen von Operatoren könnten zur Folge haben, daß bestimmte Effekte in Abhängigkeit von der jeweiligen Eingabesituation des angewandten Operators erzielt würden. Dies kann durchaus erwünscht und vorteilhaft sein, würde aber das Planen gemäß Erfüllungskriterium zur Glückssache oder zum Scheitern verurteilen. Die zusätzliche Möglichkeit, bestimmte Änderungen deduktiv zu ermitteln, hätte dieselbe Wirkung.

Des weiteren muß ausgeschlossen werden können, daß ein Merkmal nur aus der Anwendung mehrerer Operatoren resultiert. Dies wird wiederum durch die atomare Form der Merkmale garantiert. Könnte nicht davon ausgegangen werden, so müßten z.B. zur Wiederherstellung eines zerstörten Teilziels u.U. mehrere Operatoren als Retter bestimmt werden.

Schließlich sind auch die verwendeten Constraints nicht so mächtig in ihrer Ausdruckskraft, wie dies bei anderen Planungssystemen wie z.B. MOLGEN der Fall ist. Das Setzen von Constraints zur sukzessiven Spezifizierung des erwünschten Lösungsplans und damit zur schrittweisen Eingrenzung des Suchraums möglicher Pläne stellt bei TWEAK ein zentrales Hilfsmittel zur Realisierung des nichtlinearen Planungsansatzes dar. Dennoch erlauben die verwendeten Wert-Constraints nur eine grobe Wert-Spezifizierung von im Plan enthaltenen Variablen, eben Gleichheit oder Ungleichheit mit anderen Konstanten oder Variablen im Plan. Besser, aber wieder mit obig bereits mehrmals angeführtem Nachteil verbunden, wären Constraints, die flexiblere Einschränkungen der zulässigen Variablenwerte erlaubten, etwa der Art, daß auch Wertemengen oder symbolische Ausdrücke entlang von Variablen innerhalb eines Plans propagiert werden könnten. Die Darstellung der partiellen Operator-Ordnung mittels Zeit-Constraints erweist sich als sehr nützlich, die Namensgebung "Zeit-Constraints" ist nur ein wenig unglücklich gewählt, da sie den Eindruck erwecken könnte, TWEAK sei in der Lage, die in Plänen schwierig zu handhabende Größe Zeit darzustellen und bei der Planung zu berücksichtigen, was nicht der Fall ist.

Der Preis, den man für TWEAKs korrekte und vollständige Arbeitsweise bezahlen muß, sind u.U. lange Berechnungszeiten und sehr hoher Speicherbedarf für exponentiell anwachsende Suchräume und vor allem die Einschränkung, nur in sehr einfachen Domänen operieren zu können.

Kapitel 4: FKNLP - implementiertes Planungssystem

Nachdem in den vorigen Kapiteln die Vorzüge und Nachteile des nichtlinearen Planens aufgezeigt wurden und eine Beschreibung des als Vorlage dienenden Planers TWEAK erfolgte, soll nun das im Rahmen dieser Diplomarbeit von mir in COMMON LISP implementierte nichtlineare Planungssystem FKNLP vorgestellt werden. Dieses Akronym FKNLP steht übrigens für 'Friedemann Kienzler NichtLinearer Planer'.

Es erfolgt nun aber nicht eine vollständige Beschreibung des implementierten Planers, da das zugrunde liegende Konzept und die zentralen Planungsaktivitäten bereits im vorausgegangenen Kapitel über TWEAK dargelegt wurden. Vielmehr sollen, um Wiederholungen zu vermeiden, die Charakteristika und Unterschiede von FKNLP gegenüber TWEAK im Mittelpunkt der nun folgenden Betrachtungen stehen. Des weiteren wird auf die nicht nur implementierungsspezifischen Fragen, die beim Studieren der Beschreibung von TWEAK offen geblieben sind, eine Antwort mit Bezug auf das konkret erstellte Planungsprogramm gegeben, wie etwa der Aufbau der verwendeten Datenstrukturen, die Realisierung der Breitensuch-Strategie auf oberster Kontrollebene oder die programmiertechnische Umsetzung der auftretenden Indeterminismen.

4.1 Architektur des Planers

Das implementierte Planungsprogramm besitzt einen modularen Aufbau. Die einzelnen Komponenten, Module, stellen jeweils logische Einheiten von Funktionen dar, die der Lösung bestimmter Aufgaben dienen. Die während der Plangenerierung anfallenden Tätigkeiten untergliedern sich wie folgt:

- Kontrolle über den Planungsprozeß
  (z.B. Verwaltung der Plan-Warteschlange, Bestimmung des jeweils als nächstes zu betrachtenden Teilziels),
- allgemeine Planungsaufgaben
  (z.B. Erzeuger-Suche, Constraint-Propagierung, Erzeugung des Initialplans),
- Erkennen und Auflösen von Konflikten
  (z.B. Zerstörer-Suche, Konflikt-Auflösung durch Verlegung des Zerstörers),
- Kopieren von Plänen zum Aufbau der Warteschlange
  (z.B. Kopieren der beteiligten Strukturen (Planknoten und Constraints), Aktualisieren der Verweise),
- Ausgabe eines Plans während des Planungsprozesses oder zur Ergebnisdarstellung
  (z.B. Ausgabe der Variablen- und Merkmalbindungen, Ausgabe der Start- und Zielsituation).

Wegen der fehlenden Verfügbarkeit moduldefinierender Konstrukte der verwendeten Programmiersprache erfolgte diese Einteilung der beteiligten Funktionen auf logischer Ebene, d.h. die über 100 Prozeduren, aus denen sich der Planer zusammensetzt, könnten ebenso nach anderen Kriterien strukturiert werden, da das Geheimnisprinzip nicht verwirklicht ist. Die vorgenommenen Funktionszusammenfassungen scheinen allerdings sinnvoll gewählt, um für etwaige spätere Erweiterungen des Planers, beispielsweise Miteinbringung von Vorgehenswissen in die Kontrollebene, grundlegende organisatorische Modifikationen vermeiden zu können.

Zwischen den einzelnen Moduln werden in begrenztem Maße Funktionen ausgetauscht. Abbildung 4.1-1 stellt die Architektur des implementierten

Planungssystems vereinfacht dar, wobei die eingezeichneten Pfeile nicht den Datenfluß symbolisieren, sondern als "exportiert Funktionen nach" zu lesen sind. Die Rolle des Benutzers sei hier der Vollständigkeit halber ebenfalls angedeutet.

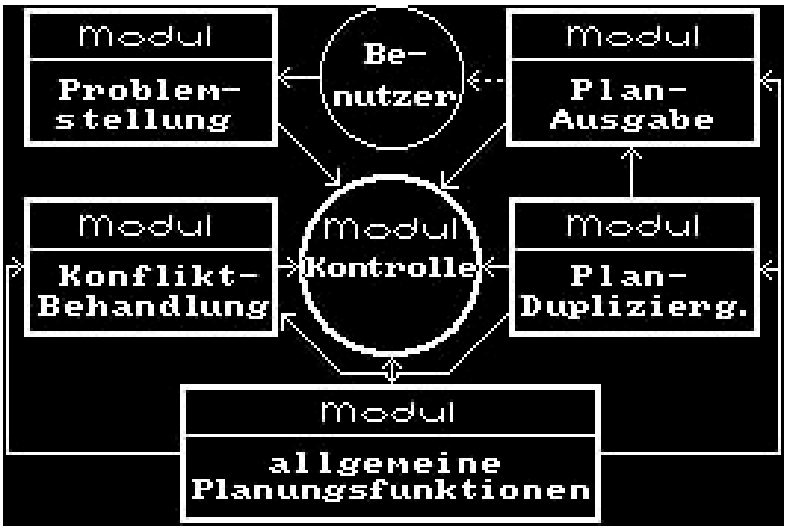


Abb. 4.1-1 Architektur des implementierten Planers FKNLP

Das gewählte Modulkonzept sorgt für eine strukturierte Darstellung sowohl des Programms an sich als auch der Erklärung desselben. Die einzelnen Teile können nacheinander betrachtet werden, was im folgenden nun auch geschehen wird.

4.1.1 Oberste Kontrollebene

Innerhalb dieses Moduls sind diejenigen Funktionen zusammengefaßt, mit denen der Planer die zentrale "Regieführung" über den Planungsprozeß bewerkstelligt. Von hier aus werden sämtliche Planungstätigkeiten angestoßen, d.h. die anderen untergeordneten Module werden aktiviert.

Die gewählten Funktionsbezeichnungen dieser Planer-Komponente (mit dem Modulnamen "Planer-Kontrolle") geben einen Hinweis auf die einzelnen Aufgaben, die auf dieser Ebene zu erfüllen sind:

```
FKNLP,
PLANEN,
PLAN-ERWEITERUNG,
ERZEUGER-ERFÜLLT-MERKMAL,
KONFLIKT-LÖSUNGEN.
```

Der Planungsprozeß wird durch Aufruf der Hauptfunktion FKNLP in Gang gesetzt, nachdem durch den Benutzer die Problemstellung, d.h. verfügbare Operator-Schablonen, gegebene Start- und erwünschte Zielsituation, spezifiziert worden ist. Der darauf aufbauenden Definition einiger weniger globaler Variablen, wie z.B. VERFÜGBARE_OPERATOREN, START und ZIEL, schließt sich die Erzeugung des Initialplans an, dessen Struktur bereits beschrieben wurde: die durch die Problemstellung gegebenen Pseudo-Operatoren Start und Ziel zusammen mit einem Zeit-Constraint, welches die Vorgängerbeziehung zwischen beiden Planknoten

darstellt, bilden die einzigen Komponenten, aus denen sich dieses anfängliche Plan-Gerüst zusammensetzt, Wert-Constraints sind noch keine gesetzt.

Eine Liste bestehend lediglich aus diesem Initialplan wird als aktueller Parameter an die eigentliche Kontroll-Funktion PLANEN übergeben, welche durch Verwaltung einer Plan-Warteschlange die Breitensuche durch den Raum möglicher Pläne steuert. Es handelt sich hierbei allerdings um eine modifizierte Breitensuch-Strategie, denn die Warteschlange wird nicht strikt am linken Anfang abgearbeitet und am rechten Ende erweitert: die aktuelle Analyse mit unmittelbar nachfolgender Entfernung aus der Warteschlange richtet sich nicht unbedingt auf das erste, sondern auf das "kleinste" Element von links, wobei die Minimalität/Optimalität sich hierbei nach der Knotenanzahl des betrachteten Plans richtet. Der Ausbau der Plan-Warteschlange erfolgt aber grundsätzlich von rechts. Auf die Auswahl des aktuell zu betrachtenden Plans nach gerade beschriebenen Kriterien und den Test, ob es sich dabei bereits um einen optimalen Lösungsplan handelt, folgt die Aktualisierung der Warteschlange, d.h. Entfernung des ausgewählten Plans aus derselben, und Initiierung der Bearbeitung des bestimmten Plans bzw. die Ausgabe des Lösungsplans, falls bereits gefunden.

Innerhalb der Funktion PLAN-ERWEITERUNG, welche als Argument den aktuell betrachteten noch nicht gelösten Plan erhält, wird das als nächstes zu bearbeitende Teilziel in Form eines noch unerfüllten Vorbedingungs-Merkmals ausgewählt und zusammen mit potentiellen alten und neuen Erzeugern, wobei letztere anwendbare Operator-Schablonen darstellen, aus denen bereits Instanzen gebildet, die aber noch nicht in den Plan integriert sind, als Argumente zum Fortschreiten der Planung an die nachfolgende Prozedur übergeben.

Für jeden ermittelten potentiellen Erzeuger generiert die Funktion ERZEUGER-ERFÜLLT-MERKMAL (mindestens) einen neuen Plan, aufbauend auf dem aktuell betrachteten, mit entsprechenden Erweiterungen, die der Erfüllung des aktuellen Teilziels dienen. Falls es sich um einen neuen Erzeuger handelt, erfolgt zunächst der Einbau in den Plan. Zwischen Erzeuger-Effekt und betrachteter unerfüllter Vorbedingung (= Teilziel) wird nur dann ein Codesignation-Wert-Constraint gesetzt, wenn zwischen beiden Merkmalen nur Codesignierbarkeit, aber noch keine Codesigniertheit besteht. Beim Erzeuger wird anschließend ein Vermerk gemacht, welches Merkmal er für welche andere Operator-Instanz erzeugt. Bei Einbau einer neuen erzeugenden Operator-Instanz muß noch überprüft werden, ob andere bestehende Erzeuger-Nutzer-Abhängigkeiten im Plan dadurch störend beeinflußt werden; ist dies der Fall, so erfolgt eine entsprechende Markierung innerhalb des Abhängigkeitseintrags des betroffenen Nutzer-Knotens, damit in einer späteren Planungsphase diesem Effekt entgegenwirkende Maßnahmen getroffen werden. Die Liste der resultierenden Pläne wird an die aufrufende Funktion als Wert zurückgeliefert, wodurch die Kontrollschleife dieser oberen Planungsebene erneut durchlaufen wird.

Schließlich besteht die Notwendigkeit, daß jeder in obig beschriebener Funktion ERZEUGER-ERFÜLLT-MERKMAL generierte Plan noch daraufhin untersucht wird, ob die Gültigkeit des betrachteten Teilziels nun auch notwendigerweise gegeben ist oder ob auf Grund störender Interaktionen irgendwelche Konflikte bestehen, die möglicherweise eine Nicht-Gültigkeit des Teilziels an dieser Stelle zur Folge hätten. Mit Hilfe der übergebenen Argumente generierter Plan, Nutzer, Teilziel und Erzeuger geht die Funktion KONFLIKT-LÖSUNGEN so vor, daß zunächst nach Zerstörern Ausschau gehalten wird. Liegen bezüglich der betrachteten Abhängigkeit keine vor, dann wird der bestimmte Erzeuger etabliert, d.h. er wird als solcher beim entsprechenden Nutzer in die Abhängigkeitsliste eingetragen, ein Zeichen dafür, daß die Gültigkeit des Teilziels notwendigerweise gegeben ist. Ist auch nur eine als Zerstörer wirkende Operator-Instanz vorhanden, analysiert der Planer zunächst die Konflikt-Art, um mögliche Lösungsansätze einzugrenzen (siehe dazu auch Unterabschnitt 4.3.4), und bestimmt dann indeterministisch die am besten geeignete Konfliktlösungs-Methode, d.h. alle durchführbaren Lösungsansätze werden "probiert", jeweils resultierend in einem eigenen Plan, und das Ergebnis an die aufrufende Funktion geliefert, welche die Einreihung in die Plan-Warteschlange veranlaßt.

Zum besseren Verständnis für die weiteren Schilderungen seien hier noch einmal die zentralen Aktivitäten auf der obersten Kontrollebene des Planers kurz festgehalten:

(1) Lesen der Problemstellung,
(2) Erzeugen des Initialplans,
(3) Bestimmung des aktuell zu betrachtenden Plans,
(4) Test: wenn Lösungsplan vorliegt, dann Ergebnisausgabe,
(5) Wahl des aktuell zu betrachtenden ungelösten Knotens,
(6) Wahl des aktuell zu betrachtenden Teilziels,
(7) Bestimmung alter und neuer Erzeuger,
(8) Generierung neuer Pläne mit je einem eingetragenem Erzeuger für das betrachtete Teilziel,
(9) Auflösung bestehender Konflikte, falls existent, für jeden in (8) neu generierten Plan mittels verschiedener angebrachter Lösungsmethoden,
(10) Einfügen der generierten Pläne in die Plan-Warteschlange und Fortsetzung des Planungsprozesses in (3).

4.1.2 Modul mit allgemeineren Planungsroutinen

Diese größte logische Einheit des implementierten Planers enthält alle die Funktionen, die sich nicht in erster Linie einer speziellen Planungsaufgabe widmen, wie dies etwa bei dem Ausgabe- oder Duplizierungsmodul (s.u.) der Fall ist, sondern vielmehr einen Pool von Routinen darstellen, die von verschiedenen anderen Planer-Komponenten genutzt werden.

Eine alphabetisch sortierte Namenliste der wichtigeren Funktionen dieses Moduls mit z.T. kurzer Erläuterung dessen, was gemacht wird, eröffnet einen Einblick in das breit gefächerte Aufgabenspektrum:

ABHÄNGIGKEITEN-AKTUALISIEREN
Bei Einbau eines neuen Erzeugers wird überprüft, ob andere im betrachteten Plan bereits bestehende Erzeuger-Nutzer-Abhängigkeiten durch die neue Operator-Instanz gestört werden. Falls dies der Fall ist, erfolgt eine Markierung des betroffenen Teilziels beim entsprechenden Nutzer.

ABHÄNGIGKEITEN-PRÜFEN

ALTER
Bestimmung des Alters einer Operator-Instanz: Je länger der Einbau-Zeitpunkt des Opertors in den betrachteten Plan zurückliegt, desto kleiner ist die als Wert gelieferte natürliche Zahl.

ANZAHL-ARGUMENTE
ANZAHL-CODESIGNIERTER-ARGUMENTE

ARGUMENTE-CODESIGNIERT
Es wird überprüft, ob die beiden an die formalen Parameter übergebenen Merkmal-Argumente A_1 und A_2 (mit Angabe der zugehörigen Operator-Instanz zur eindeutigen Kennung) miteinander codesignieren.

ARGUMENTE-NONCODESIGNIERT

BESTIMME-ACOD-HÜLLE
(= Bestimme Argumente-Codesignations-Hülle)
Für die übergebenen Merkmal-Argumente (wiederum mit Angabe der zugehörigen Operator-Instanz zur eindeutigen Kennung) gilt es, die transitive Hülle bezüglich der Codesignations-Wert-Relation zu bestimmen.

BESTIMME-DIREKTE-ABINDUNGEN
(= Bestimme direkte Argument-Wert-Bindungen)
Für die übergebenen Merkmal-Argumente werden sämtliche anderen Merkmal-Argumente im Plan ermittelt, zu denen sie in $=^1$- oder $/=^1$-Wert-Relation stehen. Welche Art der Wert-Relation berücksichtigt werden soll, kann durch einen entsprechenden Parameter vom Aufrufer festgelegt werden.

BESTIMME-DIREKTE-VORGÄNGER
BESTIMME-ETABLIERTE-ABHÄNGIGKEITEN
BESTIMME-NEGIERTE-EFFEKTE
BESTIMME-NICHTNEGIERTE-EFFEKTE

BESTIMME-OPERATOREN
Auf der Basis der vom Benutzer spezifizierten Problemstellung liefert diese Funktion eine Liste der Namen sämtlicher verfügbarer Operator-Schablonen, aufsteigend sortiert nach der Anzahl der Vorbedingungen.

BESTIMME-STARTKNOTEN

BESTIMME-UNERFÜLLTE-MERKMALE
Als Funktionswert wird eine Liste aller unerfüllter Vorbedingungs-Merkmale des als Argument übergebenen ungelösten Knotens ermittelt. Eine unerfüllte Vorbedingung ist daran zu erkennen, daß bezüglich derselben noch keine oder nur eine markierte, d.h. noch nicht etablierte Abhängigkeit eingetragen ist.

BESTIMME-VARIABLEN
BESTIMME-VORGÄNGER
BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE
BESTIMME-ZIELKNOTEN
BILDE-COD-KAPAARE (= Bilde Codesignations-Knoten-Argument-
Paare)
BILDE-KAPAARE
BILDE-NONCOD-KAPAARE

CONSTRAINT-EINBAU
Zunächst wird eine neue Wert-Constraint-Struktur erzeugt, deren Einträge durch die übergebenen Argumente (Merkmalpaar und Bezugs-Planknoten) bestimmt sind. Anschließend erfolgt der Einbau des Constraints in den betrachteten Plan, genauer in seine Wert-Constraint-Liste, aber noch ohne Verweise von den neu verbundenen Operator-Instanzen.

CONSTRAINTS-ORDNEN

EINTRÄGE-AKTUALISIEREN-E
Nach der Ermittlung eines bestimmten Erzeugers für das gerade betrachtete Teilziel bedarf es einiger Aktualisierungen bei Nutzer- und Erzeuger-Knoten, z.B. Festlegen von 'Erzeuger < Nutzer', falls dies noch nicht gilt, oder Verbinden von Erzeuger-Effekt und unerfüllter Nutzer-Vorbedingung durch Codesignation-Constraint. Diese Nebeneffekte werden bei Anwendung dieser Funktion erzielt.

ENTSPRECHENDES-ARGUMENT
ERMITTLE-ERZEUGER
ERZEUGE-INITIALPLAN
ERZEUGER-ETABLIEREN
FERTIG

GELÖST
Es wird festgestellt, ob der übergebene Planknoten gelöst ist. Dies ist dann der Fall, wenn bezüglich aller seiner Vorbedingungen Abhängigkeiten etabliert sind.

HOLE-NEUE-ERZEUGER

IST-ERZEUGER
Als Argument wird eine Operator-Instanz übergeben, die daraufhin zu untersuchen ist, ob sie ein potentieller Erzeuger für das gerade betrachtete Teilziel darstellt, wobei die bestehende Operator-Ordnung an dieser Stelle noch nicht berücksichtigt wird. Dies ist dann der Fall, wenn der betrachtete Planknoten einen erzeugenden Effekt aufweist, welcher mit dem Teilziel codesignierbar ist.

IST-NEGIERT
IST-VARIABLE
IST-VORGÄNGER

KAPAARE-ORDNEN
(= Knoten-Argument-Paare ordnen)
Es wird eine Liste von Merkmal-Argumenten (jeweils mit Angabe des Bezugs-Operators zur eindutigen Kennung) in eine Variablen- und Konstantenliste aufgespaltet.

KNOTEN-EINBAU
Der bereits erzeugte und als Argument übergebene Knoten wird in den aktuellen Plan, genauer in seine Knotenliste, eingebaut.

LOKALE-CONSTRAINTS-AUSWERTEN
Falls der betrachtete Planknoten lokale Constraints (siehe Unterabschnitt 4.2.1) definiert, wird überprüft, ob das übergebene Merkmal-Argument dadurch eine Wert-Einschränkung erfährt, und ein entsprechender Wert zurückgeliefert.

MATCH-MÖGLICH

MERKMALE-CODESIGNIERBAR
Es wird überprüft, ob die beiden übergebenen Merkmale miteinander codesignierbar sind.

MERKMALE-NONCODESIGNIERBAR

NUTZER-ABHÄNGIGKEIT-EINTRAGEN
Bei Auswertung dieser Funktion erfolgt als Nebeneffekt eine Aktualisierung der Abhängigkeitsliste des gerade betrachteten ungelösten Planknotens: sind noch irgendwelche Zerstörer im Plan vorhanden, so wird der bestimmte Erzeuger erst vorgemerkt, d.h. die Abhängigkeit entsprechend markiert; bestehen keinerlei Konflikte mehr bezüglich der aktuellen Erzeuger-Nutzer-Abhängigkeit, so findet die Etablierung des Erzeugers statt, d.h. die Abhängigkeit wird beim Nutzer als solche eingetragen bzw. deren Markierung gelöscht.

SUCHE-ALTE-ERZEUGER
SUCHE-UNGELÖSTE-KNOTEN
WÄHLE-PLAN
WÄHLE-UNERFÜLLTES-MERKMAL
WÄHLE-UNGELÖSTEN-KNOTEN

ZEITCONSTRAINTNETZ-OPTIMIEREN
Mit dem Einbau eines neuen Knotens (neuer Erzeuger oder Retter) tritt u.a. eine Einbindung desselben in die bestehende Operator-Ordnung ein. Dadurch entstehen eventuell redundante Vorgängerbeziehungen. Derselbe Effekt kann sich nicht nur bei einer Ordnungs-Erweiterung, sondern auch bei einer -Verschärfung ergeben. Bsp.: Es gelte $Op_3 < Op_1$ und $Op_2 < Op_1$. Nach Setzen eines entsprechenden Constraints ergäbe sich $Op_3 < Op_2$. Auf Grund der Transitivität der Vorgänger-Relation folgt daraus $Op_3 <^2 Op_1$, d.h. Op_3 ist auch Vorgänger von Op_1, und das Constraint bzw. der Eintrag $Op_3 < Op_1$ kann gestrichen werden. Diese Funktion entfernt solche redundanten Vorgängerbeziehungen aus der Planrepräsentation.

4.1.3 Modul zur Konflikt-Behandlung

Die Funktionen dieser Planer-Komponente untergliedern sich in drei Gruppen, die gewissermaßen mit Submodulen gleichzusetzen sind. Die erste Gruppe dient der Erkennung von Konflikten. Innerhalb der zweiten Gruppe sind alle die Routinen enthalten, welche eine Umsetzung der einzelnen Konfliktlösungs-Methoden (siehe Unterabschnitt 3.3.3) in den TWEAK spezifischen Formalismus darstellen. Schließlich können eine Reihe von Funktionen, die sekundäre Konflikt-Behandlungsaufgaben zu meistern haben, in eine dritte Gruppe gepackt werden. Um die Beschreibung nicht zu sehr in unwichtige Details abschweifen zu lassen, seien hier nur die beiden ersten Submodule erläutert. Die übrigen Funktionen können dem kommentierten beiliegenden Quellcode-Listing entnommen werden.

Routinen zur Konflikt-Erkennung

Die Planungsfunktionen mit der Aufgabe des Aufspürens von Konflikt-Situationen ähneln zum Teil in ihrer Arbeitsweise denen des im vorigen Unterabschnitt beschriebenen allgemeineren Moduls, was sich auch in der Namensgebung bemerkbar gemacht hat:

```
IST-ZERSTÖRER,
RETTER-NACHGESCHALTET,
SUCHE-ZERSTÖRER,
KONFLIKT-ANALYSIEREN.
```

IST-ZERSTÖRER ist ebenfalls wie IST-ERZEUGER eine Prädikatsfunktion, nur daß nicht im Hinblick auf Erzeugung, sondern auf Löschung geprüft wird. Als Argument wird eine Operator-Instanz übergeben, die daraufhin zu untersuchen ist, ob sie ein Zerstörer für das gerade betrachtete Teilziel darstellt, wobei die bestehende Operator-Ordnung an dieser Stelle nicht berücksichtigt wird. Der Konflikt ist dann gegeben bzw. der übergebene Planknoten wirkt dann als Zerstörer, wenn die folgenden beiden Bedingungen gelten:
- Die betrachtete Operator-Instanz weist einen löschenden Effekt auf, welcher mit dem aktuellen Teilziel codesignierbar ist.
- Die möglicherweise zerstörerische Wirkung wird nicht durch eine als Retter fungierende Operator-Instanz neutralisiert, die durch die bestehende Ordnung zwischen Zerstörer und Nutzer positioniert ist.

Um festzustellen, ob die letztere der beiden obigen Bedingungen erfüllt ist, wird, worauf der Name schon hindeutet, bei gegebener Codesignierbarkeit zwischen löschendem Zerstörer-Effekt und betrachtetem Teilziel die Funktion RETTER-NACHGESCHALTET aufgerufen. Der Test gestaltet sich dann so, daß zunächst abgefragt wird, ob löschender Zerstörer- und erzeugender Retter-Effekt in Codesignations-Wert-Relation zueinander stehen. Eine Untersuchung der bestehenden Operator-Ordnung schließt sich dann an; damit der Retter als solcher wirken kann, muß gelten:
Zerstörer $<_n$ Retter und Retter $<_m$ Nutzer für beliebige natürliche Zahlen n, m.

Die eigentliche Konflikt-Suche erfolgt durch Aufruf von SUCHE-ZERSTÖRER. Diese Funktion gibt Alarm, sobald die erste störende Interaktion bezüglich der betrachteten Teilziel-Erfüllung erkannt ist. Eine Konflikt-Situation liegt dann vor, wenn eine Operator-Instanz, die weder Nutzer noch Erzeuger noch Vorgänger des Erzeugers ist und zu deren Vorgängern ihrerseits nicht der Nutzer zählt, den Test durch IST-ZERSTÖRER mit positivem Ergebnis abschließt. Sobald ein Konflikt lokalisiert ist, wird die Suche nach weiteren abgebrochen und eben dieser Konflikt an die aufrufende Funktion zurückgemeldet.

Steht fest, daß ein Konflikt vorliegt, dann müssen Maßnahmen dagegen eingeleitet werden. Zu diesem Zweck ist es von Vorteil, zunächst durch Aufruf der Funktion KONFLIKT-ANALYSIEREN mit den Argumenten Erzeuger, Zerstörer und

Nutzer die Art des Konflikts zu bestimmen, um von vorneherein nicht anwendbare Lösungsansätze auszuschließen; dazu mehr im übernächsten Abschnitt 4.3.

Routinen zur Konflikt-Auflösung

Nachdem die verschiedenen Konfliktlösungs-Methoden bei dem gewählten Planungsansatz im vorigen Kapitel eingehend dargelegt worden sind, erübrigt sich eine weitere Erklärung der einzelnen in dieser Gruppe zusammengefaßten Funktionen, die aber dennoch hier mit Bezug auf die in [Chapman 1987] gewählten Bezeichnungen genannt sein sollen:

ORDNUNG-VERSCHÄRFEN
Dies entspricht der Verlegung des Zerstörers vor die erzeugende (-> demotion) oder hinter die nutzende (-> promotion) Operator-Instanz. Die Art der Verlegung kann durch einen ensprechenden Parameter bestimmt werden. Vor Verlassen der Funktion wird die resultierende Operator-Ordnung auf redundante Beziehungen untersucht.

SEPARIEREN
Durch Einbau eines Noncodesignation-Constraints zwischen Zerstörer-Effekt und gefährdetem Teilziel wird der Konflikt behoben (-> separation).

RETTER-EINSCHIEBEN
Diese Methode (-> white knight) bringt den größten Aufwand mit sich, was sich auch in mehreren Subroutinen bemerkbar macht, die dieser Konflikt-Behebungs-Funktion untergeordnet sind: ERMITTLE-RETTER und RETTER-NEUTRALISIERT-ZERSTÖRER, um nur die wichtigsten zu nennen.

Der Aufbau obiger drei zentraler Funktionen innerhalb dieses Moduls sieht zunächst das Herstellen einer oder im Falle des Retter-Einschiebens eventuell mehrerer Kopie(n) des betrachteten Plans vor, also noch mit Konflikt. Dem schließt sich eine spezifisch geartete Maßnahme an, sei es das Verschärfen der Operator-Ordnung, das Setzen eines Wert-Constraints oder das Einfügen eines Knotens. Schließlich bedarf es noch entsprechender Aktualisierungen und eines abschließenden Tests, ob noch weitere Zerstörer bezüglich der betrachteten Erzeuger-Nutzer-Abhängigkeit vorliegen, welche dann gegebenenfalls in späteren Planungsphasen neutralisiert werden müssen.

4.1.4 Modul zur Plan-Duplizierung

Die Notwendigkeit der Funktionen dieses Moduls ergibt sich aus der verwendeten Strategie der Breitensuche durch den Raum möglicher Pläne. Bieten sich während des Planungsprozesses an einer Stelle mehrere Möglichkeiten des weiteren Fortschreitens an, z.B. bei der Bestimmung unterschiedlicher Erzeuger oder bei der Konflikt-Auflösung mittels verschiedener Methoden, und wirkt sich die Wahl auf den weiteren Werdegang bei der Planung aus, so wird der Backtracking vermeidende, Vollständigkeit unterstützende, aber auch sehr speicher- und u.U. auch laufzeitaufwendige Ansatz gewählt, für jede Alternative einen eigenen Plan zu generieren, der zunächst durch Duplizierung aus dem aktuellen gewonnen wird.
Auf Grund der Struktur der Plan-Komponenten, Operatoren und Constraints, gestaltet sich die Kopie eines Plans - als Grundlage zum nachfolgenden unterschiedlichen Modifizieren bzw. Erweitern - etwas umständlich, denn sämtliche Verweise auf Planknoten und Constraints dürfen nicht wie die zugrunde liegenden Datenstrukturen kopiert, sondern müssen auf die neu angelegten Kopien von Planknoten und Constraints gerichtet werden.

Die Vorgehensweise beim Kopieren eines Plans läßt sich an Hand der innerhalb dieses recht kompakten Moduls integrierten Funktionen erläutern:

PLAN-KOPIEREN
Angestoßen wird der gesamte Kopiervorgang durch Aufruf dieser Funktion, welche zunächst von den anderen Routinen (s.u.) alle zum Aufbau der Plankopie notwendigen Komponenten duplizieren und die entsprechenden Einträge auf die neuen Objekte verweisen läßt, bevor sie dann aus diesen neu gewonnenen Bausteinen eine neue Planstruktur aufbaut.

KNOTENLISTE-KOPIEREN, KNOTEN-KOPIEREN
Durch wiederholten Aufruf von KNOTEN-KOPIEREN, ausgehend von der Funktion KNOTENLISTE-KOPIEREN, wird die Knotenliste des zu kopierenden Planes rekursiv abgearbeitet, sprich kopiert, indem für jeden Knoten zuerst eine neue Struktur erzeugt wird, deren Felder im Anschluß mit den entsprechenden Werten der Vorlage versehen werden, wobei Verweis-Einträge vorläufig noch unbesetzt bleiben.

CONSTRAINTLISTE-KOPIEREN, CONSTRAINT-KOPIEREN
Die Kopie der Wert-Constraintliste erfolgt analog rekursiv zu der der Knotenliste. Auch hier werden die Pointer-Einträge vorerst mit dem Default-Wert nil initialisiert.

VERWEISE-HERSTELLEN
Nachdem Knoten- und Wert-Constraintliste des ursprünglichen Plans kopiert sind, ist nur noch eine Aktualisierung sämtlicher Verweise von/auf Knoten und Constraints notwendig, wozu die beiden Funktionen VERWEISE-KOPIEREN und VERWEISE-MIT-MERKMAL-KOPIEREN ihren nützlichen Beitrag innerhalb des Duplizierungs-Moduls beisteuern.

Unmittelbar nach dem Kopiervorgang sind ursprünglicher und neuer Plan nicht voneinander zu unterscheiden. Die interne Repräsentationen der beiden sind aber voneinander verschieden, denn jeder Plan hat seinen eigenen Speicherbereich, was günstigerweise dazu führt, daß sich Änderungen bei dem einen nicht auf den anderen auswirken.

4.1.5 Modul zur Plan-Ausgabe

Die Darstellung des Lösungsplans erfolgt bei der vorliegenden Version des implementierten Planers leider nicht in graphischer Form, sondern nur rein textuell, weshalb auf eine möglichst klare und verständliche Formulierung Wert gelegt wurde.

Es wird hier auf eine Auflistung der einzelnen Funktionen verzichtet. Stattdessen werden einige Konventionen bei der Ausgabe aufgezählt, die ein klareres Erscheinungsbild des Lösungsplans erwarten lassen. Im abschließenden Kapitel 7 sind im übrigen drei Ausgabebeispiele einzusehen.

- Die Ausgabe der einzelnen Operator-Instanzen erfolgt in einer der Operator-Ordnung entsprechenden sequentiellen Reihenfolge, weshalb der Startknoten nicht immer unbedingt der erste ausgegebene Planknoten zu sein braucht; bei linearen Lösungsplänen wie bei der Sussman-Anomalie ist die aber immer der Fall. Der Zielknoten bildet bei der Ausgabe unabhängig von der bestehenden Ordnung immer das Schlußlicht bei der Ausgabe.
- Zur eindeutigen Kennzeichnung von Merkmalbindungen werden stets die Namen der entsprechenden Bezugs-Knoten in eckigen Klammern angegeben.
- Zu den Namen der einzelnen Planknoten sei angemerkt, daß derjenige Knoten, dessen Einbau in den betrachteten Plan, von Start- und Zielknoten abgesehen, am längsten zurückliegt, die Bezeichnung "KNOTEN-1" trägt. Neuere Knoten werden mit entsprechend größeren Zahlen indiziert.
- Für Wert-Bindungen von Variablen gilt folgende Konvention: liegt eine Variableninstantiierung mit einer Konstanten vor, so werden alle übrigen Variablen, die mit der betreffenden Variablen codesignieren, und alle

übrigen Konstanten und Variablen, die mit derselben noncodesignieren, - aber nur bei der Darstellung des Bezugs-Knotens der instantiierten Variablen - unterschlagen, da sie nicht mehr von Interesse sind. Ansonsten werden sämtliche in Wert-Relation zu einem Operator-Argument stehenden anderen Argumente wiederum mit Angabe des Bezugs-Planknotens zur eindeutigen Kennzeichnung angegeben.

- Wird eine Lösung gefunden, so bilden Start- und Zielsituation des gestellten Problems den Kopf der Ergebnis-Ausgabe. Liegt ein unlösbares Problem vor, ist z.B. ein Merkmal zu erreichen, das keiner der verfügbaren Operatoren als Nachbedingung besitzt, so erfolgt eine entsprechende Meldung.

4.1.6 Definition der Problemstellung

Die bisher im Rahmen dieses Abschnitts 4.1 besprochenen Module bilden den eigentlichen Planer, der domänen-unabhängig konzipiert ist. Eine Komponente aus Abbildung 4.1-1 wurde noch nicht erläutert; es handelt sich hierbei um die eigentliche Problemstellung, die für jedes Problem und für jede Domäne durch den Benutzer eingegeben werden muß. Die Definition des zu lösenden Problems gliedert sich in zwei Teile:

- gegebene Start- und erwünschte Zielsituation,
- verfügbare Operator-Schablonen.

Die Spezifikation erfolgt ausnahmslos durch Angabe einer Liste von Ausdrücken, welche die Merkmale der Start- oder Zielsituation oder die Vor- oder Nachbedingungen von Operator-Schablonen darstellen. Die Funktionen, welche die einzelnen Merkmale in eine entsprechende Operator-Struktur einflechten, tragen die Bezeichnungen STARTOPERATOR, ZIELOPERATOR und OPERATOREN.

Start- und Zielsituation stellen die Umgebung dar, innerhalb der die Planerstellung voranschreiten soll; sie bilden sozusagen den äußeren Rahmen für den Planungsprozeß. Die einzelnen Operator-Schablonen dienen als einziges Lösungshilfsmittel.

Beispiele für Spezifikationen von Problemstellungen sind wiederum Kapitel 7 zu entnehmen.

4.2 Aktionsrepräsentation und Plan-Aufbau

In der Beschreibung von TWEAK fehlen nähere Angaben darüber, wie Aktionen und Constraints als Komponenten eines Plans programmiersprachlich formuliert sind. Dies geschah wahrscheinlich mit der bewußten Absicht, den Artikel nicht zu implementierungsspezifisch mit Bezug auf das Programm TWEAK zu gestalten, sondern eher Betonung auf den eingeschlagenen Planungsweg und die dahinter stehende Idee zu legen. Chapman bewegt sich diesbezüglich auf einem eher abstrakten Level und gibt nur rudimentäre Hinweise, die Operator-Darstellung mit Vor- und Nachbedingungen und die Repräsentation eines Merkmals an sich betreffend.

Dieser Abschnitt 4.2 soll einen Einblick in die von mir gewählten Datenstrukturen zur Aktions- und Planrepräsentation vermitteln. Die Implementierungsfreiheiten diesbezüglich waren groß und ließen auch die Entwicklung von FKNLP wiederholtermaßen Rückwärtsschleifen im Erstellungszyklus durchlaufen, bis die vorliegende Version entstand.

4.2.1 Datenstrukturen

Als Darstellungsart eignet sich ein an das Frame- oder Rahmen-Konzept angelehnter Formalismus, der eine Zusammenfassung aller Aussagen über ein Objekt in einer Datenstruktur ermöglicht. Die Implementierungssprache LISP bietet die Möglichkeit, strukturierte Objekte, in LISP als Strukturen bezeichnet, mit Hilfe eines speziellen Makros zur Realisierung der Datenabstraktion einzuführen.

Eine Struktur besteht aus einer Anzahl von Feldern, die man mit den sog. Slots bei Frames vergleichen könnte, und Feldwerten oder Einträgen. Als Default- oder Initialwert erhalten bei FKNLP die meisten Felder unmittelbar nach ihrer Erzeugung lediglich den Wert nil. Das Vererbungskonzept, das ebenfalls typisch für Rahmen-Darstellungen ist, dient zur ökonomischen Datenhaltung und findet sich innerhalb des implementierten Planers bei der Anwendung verfügbarer Operator-Schablonen wieder, welche durch Eintrag des Namens der entsprechenden Operator-Schablone in das Operator-Feld umgesetzt wird; die eigentliche Operator-Struktur wird somit auch bei mehrfacher Anwendung des gleichen Operators im Plan nur einmal abgespeichert.

Nach diesen eher allgemeinen Charakteristika der gewählten Datenstrukturen schließt sich nun im folgenden eine genaue Beschreibung der Plan-Objekt-Darstellung mittels eben erwähnter Strukturen an.

4.2.2 Plan-Komponenten

Ein von FKNLP generierter Plan besteht aus einer Knoten- und einer Wert-Constraintliste. Die Listenstruktur spiegelt keinerlei Beziehungen zwischen den einzelnen Komponenten wider. Eine Plan-Struktur weist die in Abb. 4.2-1 skizzierte Gestalt auf.

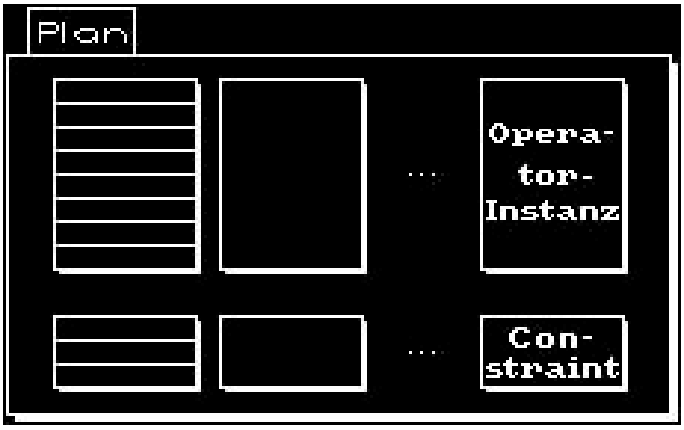


Abb. 4.2-1 Plan-Struktur

Die Komponenten eines Plans sind ihrerseits strukturierte Objekte (Abb. 4.2-2 und 4.2-3).

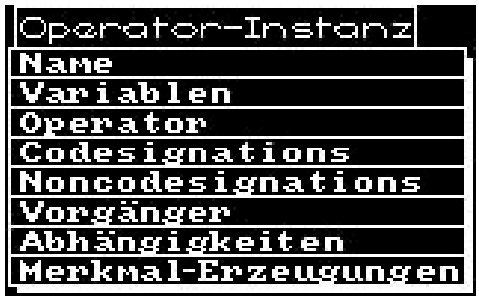


Abb. 4.2-2 Struktur eines Planknotens

Ein Planknoten, also eine im Plan eingebaute Operator-Instanz, untergliedert sich in eine Reihe von Feldern, deren Einträge die Bezüge zu den anderen Operator-Instanzen im Plan herstellen und damit das eigentliche Plangefüge erzeugen:

Feld(-name)	Feldwert
NAME	Name des Knotens zur eindeutigen Kennzeichnung im Plan; dieser Wert wird vom LISP-System generiert und ist für jeden Knoten ein anderer. Vor der Plan-Ausgabe werden diese automatisch generierten Bezeichnungen in leserlichere Namen wie "KNOTEN-2" oder "KNOTEN-START" umgeändert.
VARIABLEN	Liste der Argument-Variablen des angewandten Operators; Bsp.: (?X ?Y).
OPERATOR	Name der angewandten Operator-Schablone, dessen Wert die eigentliche Operator-Struktur mit Vor- und Nachbedingungen darstellt; Bsp.: 'STACK'.
CODESIGNATIONS	Liste mit Verweisen auf alle Wert-Constraints, die zum Zwecke der Codesignation mit einem anderen Merkmal im Plan direkt an einem der Vor- oder Nachbedingungs-Merkmale des angewandten Operators angelagert sind.
NONCODESIGNATIONS	Liste mit Verweisen auf alle Wert-Constraints, die zum Zwecke der Noncodesignation mit einem anderen Merkmal im Plan direkt an einem der Vor- oder Nachbedingungs-Merkmale des angewandten Operators angelagert sind.
VORGAENGER	Liste mit Verweisen auf alle Knoten, die direkte Vorgänger dieses Planknotens sind.
ist_abhaengig_von	Liste zweielementiger Unterlisten, deren erstes Element jeweils einen Verweis auf denjenigen Knoten im Plan bildet, der die im jeweils zweiten Listenelement genannte Vorbedingung erzeugt.
erzeugt_fuer	Liste zweielementiger Unterlisten, deren erstes Element jeweils einen Verweis auf denjenigen Knoten im Plan bildet, für den die im jeweils zweiten Listenelement genannte Nachbedingung als erzeugender Effekt für eine Vorbedingung wirkt.

Zeit-Constraints als Träger der Operator-Ordnung sind nicht als spezielle Strukturen, sondern lediglich als Verweise auf den jeweiligen Vorgänger-Knoten definiert. Die Bezeichnung Constraint ist daher etwas irreführend, da sie die Vorstellung eines eigenständigen Datengebildes nahelegt, welches ungerichtete Zusammenhänge zwischen Objekten ausdrückt. Andererseits erscheint die gewählte Namensgebung doch sinnvoll insofern, daß diese temporal constraints zum einen Relationen repräsentieren und zum andern, was für das Setzen von Constraints charakteristisch ist, den Lösungsraum entsprechend ihrer Anzahl im Plan mehr oder weniger eingrenzen.

Wert-Constraints hingegen sind als den Operator-Instanzen ebenbürtige Plan-Komponenten ausgebildet (Abb. 4.2-3).

Abb. 4.2-3 Struktur eines Wert-Constraints

Die Felder einer Wert-Constraint-Struktur ermöglichen, genau zu spezifizieren, zwischen welchen beiden Merkmalen im Plan direkte (Non-)Codesignations-Beziehung bestehen soll:

Feld(-name)	Feldwert
NAME	Name des Wert-Constraints zur eindeutigen Kennzeichnung im Plan; dieser Wert wird vom LISP-System generiert und ist für jedes Wert-Constraint ein anderer.
KNOTEN	Liste zweier Verweise auf diejenigen Knoten im Plan, deren Merkmale, spezifiziert im Feld MERKMALPAAR, durch eben dieses Constraint zueinander in (Non-) Codesignations-Wert-Relation gesetzt werden.
MERKMALPAAR	Liste zweier Merkmale, die zueinander in (Non-)Codesignations-Wert-Relation gesetzt werden.

Eine als Beispiel angeführte Darstellung eines konkreten Plans, z.B. zur Lösung des Sussman-Problems, gemäß dieser aufgezeigten internen Repräsentation wäre zu verwirrend und platzaufwendig, als daß dies dem besseren Verständnis dienen könnte. Daher genüge zur Veranschaulichung dieser kleine Ausschnitt (Abb. 4.2-4):

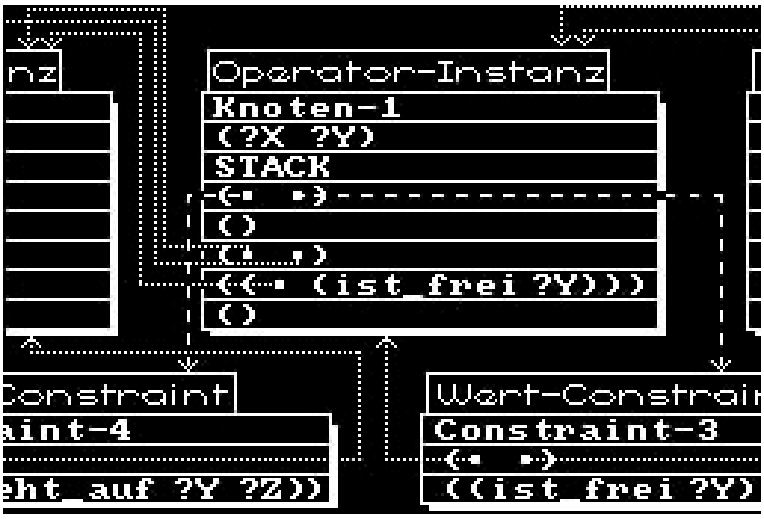


Abb. 4.2-4 Ausschnitt eines unvollständigen nichtlinearen Plans gemäß internen Repräsentation
(Gepunktete Pfeile repräsentieren Verweise auf Planknoten, gestrichelte Pfeile zeigen Verweise auf Wert-Constraints.)

Abschließend zu diesem Abschnitt sei noch der Aufbau einer Operator-Schablone aufgezeigt (Abb. 4.2-5):

Abb. 4.2-5 Struktur einer Operator-Schablone

Feld(-name)	Feldwert
VORBEDINGUNGEN	Liste von Merkmalen, die die Vorbedingungen des Operators spezifizieren, also die Merkmale, die vor Operator-Anwendung erfüllt sein müssen.
NACHBEDINGUNGEN	Liste von Merkmalen, die die Nachbedingungen des Operators spezifizieren, also die Effekte, die durch Operator-Anwendung erzeugend (nichtnegierte Merkmale) oder löschend (negierte Merkmale) wirken.
LOKALE_CONSTRAINTS	Liste dreielementiger Unterlisten der Form '(<> Argument1 Argument2)', die jeweils als "lokale" Noncodesignation-Constraints zwischen den angeführten Argumenten (Variablen oder Konstanten) des Operators wirken; lokal deshalb, weil keine Argumente anderer Operatoren im Spiel sind.

Bsp.: ((<> ?Z TISCH) (<> ?X ?Y)).

Vor dem nachfolgenden Beispiel (Abb. 4.2-6) bleibt noch anzumerken, daß lokale Noncodesignation-Constraints vor Starten des Planungsprozesses vom Benutzer einzugeben sind, sie bilden also einen Bestandteil der Problemstellung und werden nicht dynamisch während der Planung generiert, werden aber gleichermaßen berücksichtigt wie die vom Planer erzeugten Wert-Constraints.

Abb. 4.2-6 Beispiel einer Operator-Schablone aus der Blockwelt-Domäne

4.3 Planungsstrategie

Die zentralen Planungstätigkeiten FKNLPs sollen im Mittelpunkt dieses Abschnittes stehen. Dazu zählen in erster Linie das Selektieren und Erfüllen von Teilzielen, die sich im Laufe der Planung ergeben, und das Beheben von störenden Interaktionen zwischen Planästen. Von Interesse ist auch die genaue Vorgehensweise auf der obersten Kontrollebene des Planers.

4.3.1 Verwaltung der Plan-Warteschlange

Auf die beiden folgenden Fragen soll dieser Unterabschnitt eine Antwort geben:

* Woraus erklärt sich die Notwendigkeit der aufwendigen Breitensuch-Strategie?
* Welchen Sinn hat es, zur Realisierung der Breitensuche eine Warteschlange von Plänen aufzubauen und zu verwalten?

Während des Planungsprozesses treten immer wieder Situationen ein, in denen eine Entscheidung bezüglich des weiteren Fortschreitens der Planung gefällt werden muß, d.h. eine Auswahl aus mehreren Alternativen steht dann an. Dabei ist zu unterscheiden zwischen solchen Auswahlmöglichkeiten, die Konjunktionen, und solchen, die Disjunktionen darstellen.

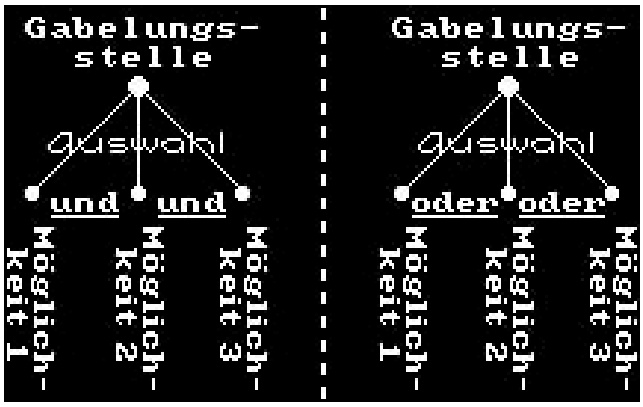


Abb. 4.3-1 Verschiedene "Gabelungsstellen" während eines Planungsprozesses (links: Konjunktion; rechts: Disjunktion)

In ersterem Fall (Abb. 4.3-1, links) müssen alle Alternativen früher oder später durchlaufen werden. Beispiele hierfür sind das Bestimmen eines zu bearbeitenden ungelösten Knotens oder eines zu erfüllenden Merkmals oder das Neutralisieren der bezüglich der gerade betrachteten Erzeuger-Nutzer-Abhängigkeit entdeckten Zerstörer, denn ein generierter Plan bildet erst dann eine Lösung, wenn alle Knoten gelöst, alle Teilziele erfüllt und alle Konflikte behoben sind. Die restriktive Aktionsrepräsentation und die auf dem Erfüllungskriterium basierende Vorgehensweise des Planers erlauben eine beliebige Bearbeitungsreihenfolge solcher anstehender Forderungen.

Der zweite angesprochene Fall (Abb. 4.3-1, rechts) liegt beispielsweise dann vor, wenn einer aus mehreren möglichen Erzeugern zu wählen ist oder wenn verschiedene Konfliktlösungs-Methoden zur Behebung einer störenden Interaktion einsetzbar wären und die am besten geeignete ausgesucht werden soll, denn für die Erfüllung eines Teilziels genügt ein Erzeuger und für die Neutralisierung eines Zerstörers ein Lösunsansatz. Hierbei würde also die Auswahl einer Alternative genügen, um die Gabelungsstelle passieren und die unmittelbar erwünschte Wirkung erzielen zu können. Um aber im nachhinein die beste Wahl getroffen zu haben, bedarf es der unabhängigen Betrachtung der einzelnen Möglichkeiten, die in unterschiedlichen Plänen resultieren, nebeneinander her, indem für jede Variante ein Plan generiert und in die Warteschlange eingereiht wird.

Zur Veranschaulichung des gerade eben Gesagten soll das folgende Beispiel dienen. Dabei wird - irgendwann während der Planung - ein Blick in die verwaltete Plan-Warteschlange unmittelbar vor und nach einem Planungsdurchlauf geworfen, bei dem die Erfüllung eines Teilziels (im Beispiel: UNERFÜLLTES_MERKMAL_1) im Mittelpunkt steht. Planauswahl, Knotenauswahl, Merkmalauswahl, Erzeugerbestimmung und Zerstörerausschaltung bilden einen Durchlauf (die vom Planer jeweils selektierten Punkte sind fett gedruckt und in Großbuchstaben geschrieben, neu generierte Pläne in Klammern angegeben):

Aufbau der Plan-Warteschlange vor dem x-ten Planungsdurchlauf:

Plan_1 - PLAN_2 - Plan_3

Ungelöste Knoten innerhalb von PLAN_2, die geringsten Abstand zum Zielknoten aufweisen (konj. Gabelstelle):

Ungelöster_Knoten_1
Ungelöster_Knoten_2
UNGELÖSTER_KNOTEN_3

Unerfüllte Vorbedingungs-Merkmale von UNGELÖSTER_KNOTEN_3 innerhalb von PLAN_2 (konj. Gabelstelle):

UNERFÜLLTES_MERKMAL_1
Unerfülltes_Merkmal_2

Mögliche alte und neue Erzeuger zur Erfüllung des betrachteten Teilziels UNERFÜLLTES_MERKMAL_1 von UNGELÖSTER_KNOTEN_3 innerhalb von PLAN_2 (disj. Gabelstelle):

ALTER_ERZEUGER_1 (--> $Plan\_2_1$)
ALTER_ERZEUGER_2 (--> $Plan\_2_2$)
NEUER_ERZEUGER_3 (--> $Plan\_2_3$)
NEUER_ERZEUGER_4 (--> $Plan\_2_4$)

Für die Abhängigkeit zwischen ALTER_ERZEUGER_1 und UNGELÖSTER_KNOTEN_3 bezüglich der Vorbedingung UNERFÜLLTES_MERKMAL_1 als Zerstörer wirkende Planknoten innerhalb von PLAN_2 (konj. Gabelstelle):

$ZERSTÖRER\_1_1$
$Zerstörer\_2_1$

Für die Abhängigkeit zwischen NEUER_ERZEUGER_4 und UNGELÖSTER_KNOTEN_3 bezüglich der Vorbedingung UNERFÜLLTES_MERKMAL_1 als Zerstörer wirkende Planknoten innerhalb von PLAN_2 (konj. Gabelstelle):

$ZERSTÖRER\_1_4$

Bezüglich der anderen Abhängigkeiten sind keinerlei Konflikte erkannt worden!

Konfliktlösungs-Methoden für die störende Interaktion zwischen ZERSTÖRER_11 und der Abhängigkeit ALTER_ERZEUGER_1-UNGELÖSTER_KNOTEN_3 bezüglich des Teilziels UNERFÜLLTES_MERKMAL_1 innerhalb von PLAN_2 (disj. Gabelstelle):

$ALTER\_RETTER\_1_1$ (--> $Plan\_2_1^1$)
$NEUER\_RETTER\_2_1$ (--> $Plan\_2_1^2$)

Konfliktlösungs-Methoden für die störende Interaktion zwischen ZERSTÖRER_14 und der Abhängigkeit NEUER_ERZEUGER_4-UNGELÖSTER_KNOTEN_3 bezüglich des Teilziels UNERFÜLLTES_MERKMAL_1 innerhalb von PLAN_2 (disj. Gabelstelle):

$SEPARIEREN\_1_4$ (--> $Plan\_2_4^1$)
$NEUER\_RETTER\_2_4$ (--> $Plan\_2_4^2$)
$NEUER\_RETTER\_3_4$ (--> $Plan\_2_4^3$)

Damit ergibt sich folgender Aufbau der Plan-Warteschlange vor dem (x+1)-ten Planungsdurchlauf):

Plan_1 - Plan_3 -
$Plan\_2_1^1$ - $Plan\_2_1^2$ - $Plan\_2_2$ - $Plan\_2_3$ - $Plan\_2_4^1$ - $Plan\_2_4^2$ - $Plan\_2_4^3$

(Tiefgesetzte Indizes geben hierbei einen Hinweis auf den für das aktuelle Teilziel gewählten Erzeuger, hochgesetzte Indizes deuten die verwendete Konfliktlösungs-Methode für einen erkannten Zerstörer an.)

Die Plan-Warteschlange, die auf der obersten Kontrollebene von FKNLP verwaltet wird, wirkt während des Planungsprozesses als Gedächtnis für die noch auf unerfüllte Teilziele hin zu untersuchenden generierten Pläne. Sie bildet eine sequentialisierte Darstellung des Suchbaums möglicher Pläne. Kennzeichen dieser Verwaltung sind, daß für einen Planungsdurchlauf jeweils derjenige Plan der Warteschlange herangezogen wird, der die wenigsten Knoten aufweist, wobei bei mehreren in Frage kommenden Kandidaten der erste von links in der Schlange selektiert wird, und im Laufe eines Durchlaufs neu generierte Pläne stets am rechten Ende der Schlange für eine nachfolgende Betrachtung angehängt werden. Es findet also eine Breitensuche statt, modifiziert dadurch, daß zur Gewährleistung der Optimalität des generierten Lösungsplans jeweils vor einem Durchlauf nicht das erste beliebige Element aus der Warteschlange gewählt wird, sondern das bezüglich Knotenanzahl kompakteste von links.

Eine Tiefensuch-Strategie scheint zunächst naheliegend, um eventuell schnellere Planungszeiten zu erhalten. Daß man bei diesem Ansatz Gefahr läuft, in eine Endlosschleife zu geraten, liegt daran, daß zur Erzeugung oder Rettung eines Merkmals im Falle eines lösbaren Problems immer auch ein neuer Operator in den bestehenden Plan eingebaut werden kann, welcher u.U. auf bestehende Abhängigkeiten selbst als Zerstörer wirkt, der seinerseits durch Einschieben eines neuen Retters neutralisiert werden kann, welcher eventuell wiederum einen Konflikt hervorruft, usw. Da FKNLP zusätzlich immer die Möglichkeit in Betracht zieht, zur Erfüllung eines Teilziels einen neuen Knoten einzubauen, muß die Plan-Warteschlange gemäß obiger Strategie verwaltet werden.

4.3.2 Bearbeiten von Teilzielen

Dieser zentrale Vorgang setzt zunächst eine Auswahl des aktuellen Teilziels aus den noch unerfüllten Merkmalen im Plan voraus. Hierfür wiederum muß ein ungelöster Planknoten selektiert werden. FKNLP sucht zur Realisierung der Rückwärtssuche innerhalb der Knotengeneration, die ungelöste Knoten aufweist und sich durch minimalen Abstand zum Zielknoten auszeichnet. Innerhalb dieser eindeutig bestimmbaren Generation wird derjenige Knoten, dessen Einbau in den aktuellen Plan am längsten zurückliegt, als zu betrachtender ungelöster Knoten deklariert. Unter seinen noch unerfüllten Vorbedingungen wird das Merkmal mit den wenigsten ungebundenen Argumentvariablen zum aktuellen Teilziel für den anschließenden Planungsdurchlauf. Mit dieser Strategie soll der Planungsprozeß in eine Richtung gelenkt werden, in der zumindest partiell bereits Information vorliegt; dem "Tappen im Dunkeln" kann dadurch ein wenig entgegengewirkt werden.

Beispiel: Der gewählte ungelöste Knoten besitze die unerfüllten Vorbedingungen
(ist_frei ?X), wobei ?X ungebunden, und
(Hand_hält ?Z), wobei ?Z =! A gilt.
[?X, ?Z bezeichnen Variablen, A ist eine Konstante.]
Dann wird das zweite Vorbedingungs-Merkmal zum aktuellen Teilziel, da bezüglich seiner Argumente bereits gewisse Informationen vorliegen.

Dieser Selektionsmechanismus wirkt sich nicht auf das Finden oder Nicht-Finden einer Lösung aus. Bestenfalls wird der Planungsprozeß dadurch etwas beschleunigt. Im vorausgegangenen Unterabschnitt wurde bereits betont, daß bei derartigen konjunktivischen Gabelstellen die Reihenfolgefestlegung beliebig ist.

Sind ungelöster Knoten (=: N) und unerfülltes Teilziel (=: M) bestimmt, beginnt die nächste Phase des Planungsdurchlaufs, die Suche nach möglichen Erzeugern. Ein Operator eignet sich dann als potentieller Erzeuger, wenn er eine erzeugende Nachbedingung besitzt, die mit M codesignierbar ist. Für den Test diesbezüglich werden zuerst eingebaute Operator-Instanzen herangezogen, bevor die verfügbaren Operator-Schablonen auf Anwendbarkeit mit erwünschter Wirkung untersucht werden. Die Codesignierbarkeit von Merkmalen wird zwischen den jeweils korrespondierenden Merkmal-Argumenten festgestellt, wobei die

transitiven Hüllen bezüglich der Wert-Relation durchsucht werden, was einer Propagierung entlang der Argumente via Noncodesignation-, Codesignation- und lokaler Wert-Constraints gleichkommt.

Bei Existenz eines bereits vorgemerkten Erzeugers werden keine weiteren betrachtet, sondern sogleich beginnt die Suche nach etwaigen Konflikten. Ein Erzeuger Ev kann in zweierlei Situationen vorgemerkt werden:

- Ev wird als erzeugende Operator-Instanz bestimmt, aber auf Grund mehrerer Konflikte erfolgt noch keine Etablierung als Erzeuger, da innerhalb eines Planungsdurchlaufs immer nur ein Zerstörer neutralisiert wird und das Nichtvorhandensein irgendwelcher diesbezüglicher Konflikte die Voraussetzung für eine Etalierung ist.

- Ev wurde in einem vorausgegangenen Durchlauf als etablierter Erzeuger eingetragen, aber neu hinzugekommene Operator-Instanzen wirken auf die Abhängigkeit N-M-Ev als Zerstörer und veranlassen eine Vormerkung von E, damit in einem nachfolgenden Planungsdurchlauf, in dem wieder M als Teilziel selektiert wird, Konfliktlösungs-Methoden eingeleitet werden.

Die Vormerkung eines Erzeugers erfolgt durch Markierung des Abhängigkeits-Eintrags beim entsprechenden Nutzer, die Etablierung durch Neueintragung in das Abhängigkeits-Feld, falls keine Vormerkung vorausging, oder durch Entfernung der Markierung beim entsprechenden Nutzer. Für die Erfüllung eines Merkmals wird immer nur ein Erzeuger eingetragen.

4.3.3 Vorgehen bei auftretenden Konflikten

Die letzte Phase eines jeden Planungsdurchlaufs hat die Aufgabe, sämtliche im Rahmen der aktuellen Teilzielerfüllung in vorausgegangenen Phasen generierten unvollständigen Pläne nach Konflikten zu durchsuchen, die der Merkmalerfüllung entgegenwirken könnten. Für den jeweils betrachteten Plan fallen dabei folgende Einzeltätigkeiten an:

Die Suche nach Zerstörern beginnt vor dem Zielknoten des jeweiligen Plans und vollzieht sich generationsweise in Richtung Startknoten. Werden keine Zerstörer gefunden, ist dies ein Zeichen dafür, daß der ermittelte Erzeuger etabliert werden kann. Sobald aber eine auf Grund der bestehenden Ordnung möglicherweise "gefährliche" Operator-Instanz entdeckt wird, die eine löschende Nachbedingung besitzt, welche mit dem aktuellen Teilziel codesignierbar ist, dann erkennt der Planer dies als Konflikt, den es zu beheben gilt, und setzt die Zerstörer-Suche nicht weiter fort. FKNLP geht dabei so vor, daß es zunächst die Konflikt-Art analysiert, um mögliche Lösungsansätze einzugrenzen, und dann indeterministisch die am besten geeignete Konfliktlösungs-Methode bestimmt.

Abhängig vom Verhältnis der drei Planknoten E (Erzeuger), N (Nutzer) und Z (Zerstörer) innerhalb der Operator-Ordnung lassen sich drei Grundformen von Konflikten unterscheiden:

(1) Parallel-Konflikt,
(2) Gabel-Konflikt,
(3) Linear-Konflikt.

Zu (1): Ein Parallel-Konflikt besteht dann, wenn für einen Zerstörer Z weder $Z <_n N$ noch $E <_n Z$, aber auch nicht $N <_n Z$ und $Z <_n E$ für beliebiges $n > 0$ gilt (Abb. 4.3-2).

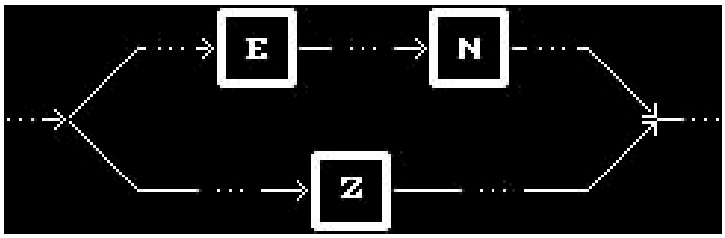


Abb. 4.3-2 Parallel-Konflikt

Mögliche Konfliktlösungs-Methoden bei dieser Konflikt-Art:
- Vorverlegung ($Z <_n E$),
- Nachverlegung ($N <_n Z$),
- evtl. Separation, falls Zerstörer-Effekt und Teilziel noncodesignierbar sind,
- Einschaltung eines Retter-Knotens.

Bei einem Parallel-Konflikt bieten sich die meisten Möglichkeiten, die störende Wirkung zu beheben.

Zu (2): Ein Gabel-Konflikt besteht dann, wenn für einen Zerstörer Z entweder $E <_n Z$ und nicht $N <_m Z$ (Abb. 4.3-3) oder $Z <_n N$ und nicht $Z <_m E$ (Abb. 4.3-4) für beliebige $n, m > 0$ gilt.

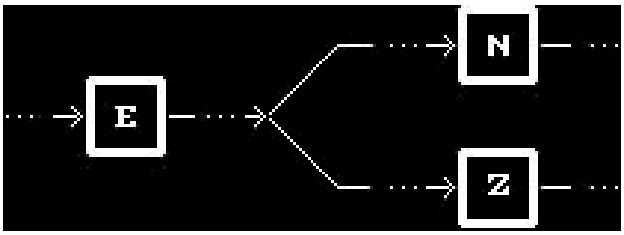


Abb. 4.3-3 Gabel-Konflikt vom Typ 1: $E <_n Z$

Mögliche Konfliktlösungs-Methoden bei dieser Konflikt-Art (Gabel-Konflikt_1):
- Nachverlegung ($N <_n Z$),
- evtl. Separation, falls Zerstörer-Effekt und Teilziel noncodesignierbar sind,
- Einschaltung eines alten oder neuen Retter-Knotens.

Eine Linearisierung derart, daß der Zerstörer vor den Erzeuger verlegt wird, ist in diesem Fall zur Operator-Ordnung nicht verträglich und daher nicht durchführbar.

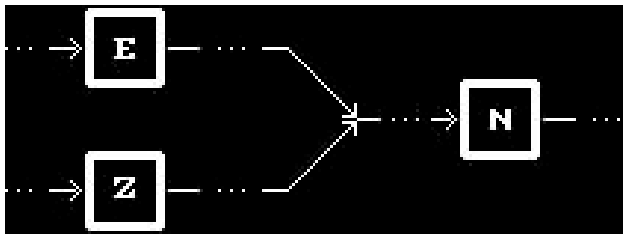


Abb. 4.3-4 Gabel-Konflikt vom Typ 2: $Z <_n N$

Mögliche Konfliktlösungs-Methoden bei dieser Konflikt-Art (Gabel-Konflikt_2):
- Vorverlegung ($Z <_n E$),
- evtl. Separation, falls Zerstörer-Effekt und Teilziel noncodesignierbar sind,
- Einschaltung eines alten oder neuen Retter-Knotens.

Eine Linearisierung derart, daß der Zerstörer hinter den Nutzer verlegt wird, ist in diesem Fall zur Operator-Ordnung nicht verträglich und daher nicht durchführbar.

Zu (3): Ein Linear-Konflikt besteht dann, wenn für einen Zerstörer Z sowohl $E <_n Z$ als auch $Z <_m N$ für beliebige $n, m > 0$ gilt (Abb. 4.3-5).

Abb. 4.3-5 Linear-Konflikt

Mögliche Konfliktlösungs-Methoden bei dieser Konflikt-Art:
- evtl. Separation, falls Zerstörer-Effekt und Teilziel noncodesignierbar sind,
- Einschaltung eines alten oder neuen Retter-Knotens.

Das Hilfsmittel der Linearisierung zur Konflikt-Auflösung entfällt hier gänzlich, da die bestehende Operator-Ordnung kein Verlegen des Zerstörers vor den Erzeuger oder hinter den Nutzer zuläßt.

Je nach Konflikt-Art werden also unterschiedliche Maßnahmen zur Behebung des Konflikts durchgeführt, welche jeweils in einem neuen, von dem erkannten Konflikt befreiten Plan resultieren. Für jeden sich so ergebenden Plan wird abschließend noch kontrolliert, ob weitere Zerstörer vorliegen. Ist dies nicht der Fall, dann kann der ermittelte Erzeuger beim Nutzer in der Abhängigkeits-liste als etabliert eingetragen werden. Wird aber doch mindestens ein weiterer Zerstörer entdeckt, muß die Etablierung verschoben werden: Der Erzeuger wird beim Nutzer erst vorgemerkt, damit das betrachtete Teilziel als noch nicht gelöst markiert ist und somit in einem nachfolgenden Planungsdurchlauf unter Berücksichtigung der Vormerkung erneut bearbeitet wird. Mit der Etablierung oder Vormerkung des Erzeugers sind die Modifikationen und Erweiterungen des gerade

betrachteten und in diesem Planungsdurchlauf erzeugten Planes abgeschlossen, so daß die Einreihung in die Plan-Warteschlange vom rechten Ende her erfolgen kann.

Wenn alle in dieser letzten Phase eines Duchlaufs erzeugten Pläne auf diese Weise behandelt worden sind, findet der Übergang in den nächsten Planungsdurchlauf statt: erneute Plan-Auswahl, Test auf Lösung, Teilziel-Bestimmung, usw. Dieser Zyklus wiederholt sich solange, bis eine Lösung gefunden ist.

4.3.4 Umsetzung des Erfüllungskriteriums

Die Vorgehensweise bei der Bearbeitung eines Teilziels, insbesondere bei der Bestimmung der Gültigkeit eines Merkmals an einer bestimmten Stelle im Plan und der erforderlichen Behebung störender Interaktionen, stützt sich bei der Planung durch FKNLP im wesentlichen auf das im vorigen Kapitel dargelegte Erfüllungskriterium.

Die einzige Abweichung ergibt sich beim Auflösen von Konflikten durch Einschieben eines Retters. Um nicht mehr als nötig festzulegen, also der Vollständigkeit wegen die schwächste Bedingung zu formulieren, wird bei TWEAK die Kopplung zwischen löschendem Zerstörer-Effekt (=: M_Z) und erzeugendem Retter-Effekt (=: M_R) durch eine Implikation via Constraints realisiert, wodurch die Bindung erst dann aufgebaut aufgebaut wird, wenn auf Grund einer bestehenden Codesigniertheit und nicht nur Codesignierbarkeit zwischen Zerstörer-Effekt und betrachtetem Teilziel (=: M) tatsächlich ein Konflikt vorliegt. Zur Umgehung der Schwierigkeiten bei der Einführung zusätzlicher dreistelliger Implikations-Constraints erfolgt bei FKNLP die Kopplung zwischen Zerstörer- und Retter-Effekt durch Äquivalenz (Abb. 4.3-6; zum Vergleich: Abb. 3.3-10 im vorigen Kapitel).

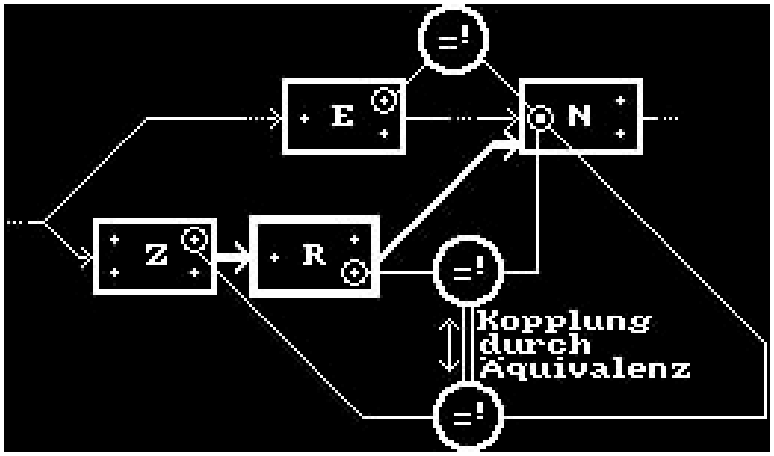


Abb. 4.3-6 Konflikt-Auflösung bei FKNLP durch Einschieben eines Retters: Kopplung zwischen Zerstörer- und Retter-Effekt durch Äquivalenz

Diese stärkere Kopplung resultiert in einer Forderung, die etwas stärker ist als notwendig, aber immer noch eine schwächere Bedingung darstellt als eine direkte Bindung von Retter-Effekt und betrachtetem Teilziel, die unabhängig von der Codesigniertheit zwischen Zerstörer-Effekt und Teilziel aufgebaut würde. Der Fall, in dem bei Äquivalenz-, aber nicht bei Implikations-Kopplung zwischen Zerstörer- und Retter-Effekt eine Codesignations-Bindung zwischen wiedergutmachendem Retter-Effekt und betrachtetem Teilziel eintritt, liegt in der zusätzlichen Implikation

$M =^{!} M_Z \;<=\; M =^{!} M_R$
(vgl. bei TWEAK: $M =^{!} M_Z \;=>\; M =^{!} M_R$)

begründet: Wenn Teilziel und Retter-Effekt codesignieren, dann überträgt sich die Codesignation nun auch auf Teilziel und Zerstörer-Effekt, was bei TWEAKs Ansatz nicht der Fall wäre. Dadurch entsteht kein inkonsistenter Zustand, nur wird entgegen der strengen Vorgehensweise gemäß Constraint- Posting-Ansatz etwas mehr festgelegt als eigentlich erforderlich.

Diese einzige Abweichung vom Prinzip der schwächsten Forderungen, das bei der Implementierung des Planers ansonsten immer als Orientierung galt, bringt keine größeren Nachteile mit sich, wenn auch von der theoretischen Vollständigkeit gewisse Abstriche gemacht werden müssen. Aber die Probleme, deren Lösungen auf Grund dieser Überbeschränkung trotz Existenz nicht gefunden werden, stellen eine nur sehr kleine Gruppe möglicher Problemstellungen dar, und es ist schwierig, ein Beispiel dafür zu finden.

Die Repräsentation der Äquivalenz

$M =^{!} M_Z \;<=>\; M =^{!} M_R$

gestaltet sich auf recht elegante Weise durch Setzen eines Codesignation-Constraints zwischen Zerstörer- und Retter-Effekt mit der Wirkung $M_Z =^{!} M_R$ (Abb. 4.3-7).

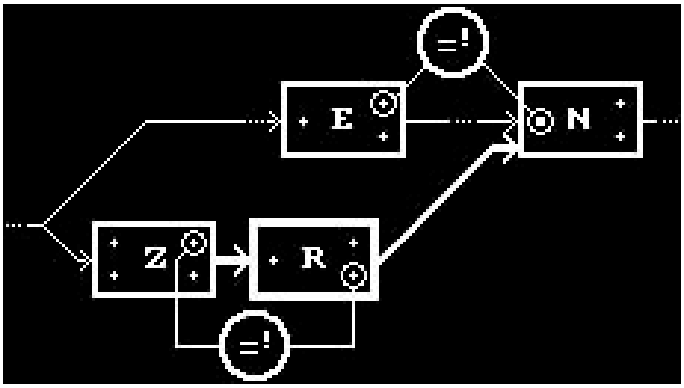


Abb. 4.3-7 Konflikt-Auflösung bei FKNLP durch Einschieben eines Retters: Löschender Zerstörer-Effekt und wieder erzeugender Retter-Effekt codesignieren mittels Constraint.

Kapitel 5: Beurteilung des gewählten Planungsansatzes

Das Grundproblem beim Planen besteht - wie bei vielen anderen Problemen in der Künstlichen Intelligenz auch - in der Komplexität des Suchraums. Die unterschiedlichen Planungstechniken, die in Kapitel 1 vorgestellt wurden, haben zwar alle ihre eigenen Charakteristika, zeichnen sich aber in ihrer Gesamtheit dadurch aus, daß sie jeweils einen mehr oder weniger guten Versuch darstellen, mit Hilfe unterschiedlicher Ansätze die Schwierigkeit der totalen Suche durch den Raum möglicher Plankonstellationen in den Griff zu bekommen. Des weiteren sollte sich ein Planer durch eine Reihe von Eigenschaften auszeichnen, die seine praktische Anwendbarkeit unter Beweis stellen. Dabei stehen Effizienz, Mächtigkeit und realistische Einsatzmöglichkeiten auf der einen Seite und Korrektheit, Vollständigkeit und Optimalität auf der anderen. Die aufgezählten Eigenschaften sind schwer in Einklang zu bringen. Es gilt, den richtigen Kompromiß bei der Wahl verschiedener Planungstechniken und Repräsentations-formalismen zu treffen, um dann bei der Findung des Lösungsweges für die gestellten Anwendungsprobleme einigermaßen erfolgreich zu sein.

Die Technik des nichtlinearen Planens, beispielsweise realisiert mit Hilfe von Constraints, wie dies bei TWEAK und FKNLP der Fall ist, weist in die richtige Richtung. Ein reines Durchprobieren sämtlicher möglicher Konstellationen wie beim linearen Planen ist auf Grund der exponentiell anwachsenden Suchräume ein unrealistischer Ansatz ohne entscheidende Verbesserungsmöglichkeiten. Informationen, die sich während des Planungsprozesses ergeben, etwa die Eingrenzung von Variablenwertebereichen, können dazu genutzt werden, mögliche Entscheidungsalternativen an Gabelungsstellen auszuschließen oder zu favorisieren. Darüber hinaus bietet dieser Ansatz den Vorteil, die Idee, vorhandenes Wissen zum Fällen von Entscheidungen einzusetzen, dahin gehend zu erweitern, zusätzliches Wissen etwa in Form von Vorgehens- oder Strategiewissen a priori in den Planungsprozess miteinzubeziehen. Formulierung und inhaltliche Auswahl solchen Wissens werfen dann weitere Probleme auf, die sich in gewissem Maße aber auch schon bei der Wahl der Aktionsrepräsentation ergeben.

Bei Planern, wie z.B. MOLGEN, SIPE, TWEAK und natürlich auch FKNLP, die den Constraint-Ansatz verfolgen, werden im Laufe der Planung immer mehr Daten gesammelt, um den Suchraum sukzessive einzugrenzen. Es wird vermieden, frühzeitige (Fehl-)Entscheidungen zu treffen, die Backtracking nach sich ziehen könnten. Der Planungsprozeß gestaltet sich als eine schrittweise Objekt-Definition: Der zu generierende Lösungsplan wird wie ein Mosaik zusammengesetzt.

Das Setzen von Constraints allein verhindert aber noch nicht, mögliche Alternativen an Verzweigungspunkten auf ein überschauberes Maß zu reduzieren. Ein weiteres wichtiges Konzept ist das Planen über meherere Abstraktionsstufen hinweg. Auch hinter diesem Ansatz des mehrstufigen Planens, den es in den Ausprägungen Situations- und Operatorabstraktion gibt, steckt wieder die Idee, zunächst nur solche Festlegungen zu treffen, die von Interesse und möglichst nicht mit der Gefahr verbunden sind, zu einem späteren Zeitpunkt samt den darauf aufbauenden Entscheidungen wieder zurückgenommen werden zu müssen. Man erkennt die Parallelen zum nichtlinearen Planen.

Der Schlüssel zum Erfolg liegt darin, die einzelnen Planungstechniken, wie dies bei MOLGEN der Fall ist, geschickt zu kombinieren und diese mittels geeigneter Heuristiken auf angebrachte Weise auszuwählen bzw. anzuwenden. Die Zielsetzungen des im Rahmen dieser Diplomarbait implementierten Planers waren eher theoretischer Natur. Der nichtlineare Planungsansatz allein eröffnet sicher nicht die Vorteile, die eine Verknüpfung mit anderen Verfahren auf unterschiedl-ichen Ebenen mit sich brächte. Aber an Hand eines rein nichtlinear und einstufig vorgehenden Planers lassen sich die spezifischen Eigenheiten besser studieren.

Die beiden Techniken des nichtlinearen Planens und des Setzens von Constraints harmonieren seht gut miteinander, da bei beiden die gleiche Strategie verfolgt wird, partielle Beschreibungen zu spezifizieren, die schrittweise zu vervollständigen sind. Für das Auflösen von Konflikten erweist sich der von TWEAK gewählte Ansatz als flexibel und systematisch anwendbar. Allerdings ist die nichtlineare Darstellung von Plänen ein "Nährboden" für derartige Konflikte, da die nur partielle Ordnung viele Freiheiten bezüglich Operator-Konstellationen offen läßt, darunter auch solche mit störenden Interaktionen untereinander.

Domänen-Unabhängigkeit ist eine Forderung, die sich nur auf den Kern des Planers beschränken, nicht aber ein Verbot für die Einführung bereichs-spezifischer Heuristiken darstellen sollte. Auch eine Vorgehensweise gemäß Erfüllungskriterium läßt die nachträgliche Miteinbringung etwa von Vorgehens-wissen zu, nur der Anspruch der Vollständigkeit geht dadurch verloren; aber was nützt eine derartige Eigenschaft, die in der Praxis an den Grenzen der Ressourcen Laufzeit und Speicherbedarf scheitert. Vollständigkeit muß auch immer in Relation zur Mächtigkeit des gewählten Formalismus betrachtet werden: Wenn komplexere Problemstellungen nicht in einer für den Planer verständlichen Form dargestellt werden können, dann nützt die Vollständigkeit bei der Lösungssuche nicht allzu viel.

Die Umsetzung der Beschreibung von TWEAK in ein lauffähiges Programm, das in der Lage ist, kleine Probleme aus der fiktiven Blockwelt-Domäne zu lösen, zeigt die grundsätzliche Verwendbarkeit des gewählten Planungsansatzes und den rationellen Gebrauch von Constraints bei dem verwendeten Formalismus. Die Suchraum-Problematik wird dabei nicht gemeistert, was aber auch nicht die Zielsetzung war. Die Planungsstrategie gemäß Erfüllungskriterium scheint vielversprechend und der Integration mit anderen Vorgehensweisen wert. Beispielsweise könnten Constraints dazu verwendet werden, mehrstufiges Planen zu realisieren und das Erfüllungskriterium auf verschiedene Abstraktionsstufen auszudehnen. Auf jeden Fall eignet sich die nichtlineare Planungstechnik zum Einbau in eine Planungs-Shell, die Kombinationen und unterschiedliche Selektionen von Strategien erlaubt. Als bereits realisiertes und in begrenztem Umfang innerhalb einer speziellen Domäne erfolgreich arbeitendes Planungssystem mit der angesprochenen Integration des nichtlinearen Planungsansatzes, des Constraint-Posting und der Operatorabstraktion ist MOLGEN zu nennen.

FKNLP kann als Vorlage für andere neue nichtlineare Planer mit anderen Schwerpunkten in der Zielsetzung und den Einsatzmöglichkeiten dienen, denn die Vorgehensweise ist gut nachvollziehbar und die Architektur modular aufgebaut, was auch eine nachträgliche Miteinbringung von Vorgehens- oder Strategiewissen erleichtert.

```
================================================
Kapitel 6:    Dokumentation: Handhabung von FKNLP
================================================
```

Nachdem die theoretischen Hintergründe und die Funktionsweise des implementierten Planers erklärt wurden, sind die nächsten beiden Kapitel eher als Anhang anzusehen, in denen das praktische Arbeiten mit FKNLP im Mittelpunkt steht.

6.1 Notwendige Eingaben

Der Lösung einer Planungsaufgabe muß natürlich die Spezifikation der Problemstellung durch den Benutzer vorausgehen. Die Eingabe erfolgt nicht nach Programmstart, sondern zusammen mit der Definition der für den Planer zur Verfügung stehenden Operatoren im Rahmen der Planungsvorbereitung vor dem eigentlichen Planungsprozeß. Leider ist die Benutzerschnittstelle nicht sehr komfortabel gestaltet: Die notwendigen Eingabegrößen, die im folgenden genauer aufgeschlüsselt werden, holt sich der Planer nicht von der Standard-Eingabe, sondern aus einer Datei mit dem Namen "Problemstellung.lisp" im aktuellen Directory. Der Benutzer muß also über einen Editor diese Eingabedatei entsprechend der zu lösenden Aufgabe modifizieren.

6.1.1 Problemspezifikation

Ein Teil der Problemstellung bildet die Spezifikation der gegebenen Start- und der erwünschten Zielsituation. Innerhalb der Datei "Problemstellung.lisp" sind hierfür die beiden Prozeduren 'STARTOPERATOR' und 'ZIELOPERATOR' zuständig, denn deren Ausführung zu Beginn der Planung unmittelbar nach dem Programmstart führt zur Definition der beiden Pseudo-Operatoren 'START' und 'ZIEL', welche als Werte der beiden Funktionen zurückgeliefert werden. Sie bilden die wesentlichen Bestandteile des Initialplans, der das Gerüst einer jeden Planung mit FKNLP darstellt.

Die Situationsbeschreibungen setzen sich jeweils aus einer Liste von Merkmalen zusammen.

Am Beispiel der Formulierung des Sussman-Problems soll die genaue Form der notwendigen Eingaben aufgezeigt werden. Im nachfolgenden Auszug aus der Datei "Problemstellung.lisp", die übrigens in ihrer Gesamtheit auf den letzten drei Seiten des FKNLP-Sourcecode-Listings (siehe Anhang) abgebildet ist, sind zur Verdeutlichung die vom Benutzer durchzuführenden Programm-Modifikationen fett und vergrößert gedruckt:

```
(Konvention für die Eingabe:
   - Variablen werden als solche durch vorangestelltes Fragezeichen kenntlich
     gemacht, z.B. '?X'.
   - Der Wert NIL nimmt insofern eine Sonderstellung ein, daß er weder als
     Konstante noch als Variable behandelt wird; eine Variable kann auch nicht
     mit diesem Wert instantiiert werden.)


; ACHTUNG:
; -------
; Hier ist in den unten gekennzeichneten Bereichen (zwischen '*'-Zeilen)
; fuer das jeweilige Planungsproblem eine entsprechende Modifikation
; durch den Benutzer durchzufuehren !!
```

```
; Stand: 29.05.1990


; ========================================================================
;                          Problemspezifikation
;                          --------------------
; Beschreibung der gegebenen Startsituation und der erwuenschten Ziel-
; situation dadurch, dass die jeweils geltenden Merkmale aufgelistet
; werden !
; ========================================================================

  (DEFUN STARTOPERATOR ()
        (MAKE-OPERATOR
             :NACHBEDINGUNGEN
                      '(
; ------------------------------------------------------------------------
; Im folgenden sind die Merkmale der gegebenen STARTSITUATION anzugeben:
; ************************************************************************
; *                                                                      *
      (steht_auf C A)
      (steht_auf A TISCH)
      (steht_auf B TISCH)
      (ist_frei C)
      (ist_frei B)
; *                                                                      *
; ************************************************************************
                       )
              :LOKALE_CONSTRAINTS
                      '(
; ------------------------------------------------------------------------
; Im folgenden sind die lokalen Wert-Constraints, die in der
; STARTSITUATION Gueltigkeit haben, anzugeben:
; ************************************************************************
; *                                                                      *

; *                                                                      *
; ************************************************************************
                       )))



  (DEFUN ZIELOPERATOR ()
        (MAKE-OPERATOR
             :VORBEDINGUNGEN
                      '(
; ------------------------------------------------------------------------
; Im folgenden sind die Merkmale der erwuenschten FINALSITUATION anzu-
; geben:
; ************************************************************************
; *                                                                      *
      (steht_auf A B)
      (steht_auf B C)
; *                                                                      *
; ************************************************************************
                       )
             :LOKALE_CONSTRAINTS
                      '(
; ------------------------------------------------------------------------
; Im folgenden sind die lokalen Wert-Constraints, die in der
```

```
; FINALSITUATION Gueltigkeit haben, anzugeben:
; ************************************************************************
; *                                                                      *

; *                                                                      *
; ************************************************************************
                        )))
```

Die Definition lokaler Constraints bei Start- oder Zieloperator gibt nur dann einen Sinn bzw. kann von Interesse sein, wenn mindestens eines der Merkmale von Start- oder Zielsituation Variablen als Argumente von Prädikaten enthält.

6.1.2 Verfügbare Operator-Schablonen

Die Definition der verfügbaren Operator-Schablonen bildet den zweiten Teil der Problemstellung. Die vom Planer angestoßene Auswertung der Prozedur 'OPERATOREN' liefert als Wert eine Liste der Namen sämtlicher verfügbarer Operator-Schablonen, deren Werte ihrerseits die eigentlichen Operator-Strukturen mit Vor- und Nachbedingungen und evtl. lokalen Constraints darstellen, z.B. (NEWTOWER PUTON).

Die Fortsetzung des Auszugs aus der Datei "Problemstellung.lisp" mit der Formulierung des Sussman-Problems soll beispielhaft wiederum die Stellen hervorheben, an der der Benutzer die Merkmale zur Definition der einzelnen Operatoren vor dem Programmstart eintragen muß.

```
; ======================================================================
;                        Operatorspezifikation
;                        ---------------------
;     Beschreibung der einzelnen dem Planer zur Verfuegung stehenden
;     Planungsoperatoren (die Reihenfolge spielt keine Rolle) !
; ======================================================================

  (DEFUN OPERATOREN ()
       '(
; ----------------------------------------------------------------------
; An dieser Stelle ist die OPERATORBEZEICHNUNG des ersten Operators
; anzugeben:
; ************************************************************************
; *                                                                      *
     PUTON
; *                                                                      *
; ************************************************************************
              (MAKE-OPERATOR :VORBEDINGUNGEN
                             '(
; ----------------------------------------------------------------------
; Im folgenden sind die VORBEDINGUNGEN des ersten Operators anzu-
; geben:
; ************************************************************************
; *                                                                      *
     (steht_auf ?X ?Z)
     (ist_frei ?X)
     (ist_frei ?Y)
; *                                                                      *
; ************************************************************************
                              )
```

```
                              :NACHBEDINGUNGEN
                               '(
; ---------------------------------------------------------------------
; Im folgenden sind die NACHBEDINGUNGEN des ersten Operators anzu-
; geben:
; *********************************************************************
; *                                                                   *
     (steht_auf ?X ?Y)
     (ist_frei ?Z)
     (NICHT (steht_auf ?X ?Z))
     (NICHT (ist_frei ?Y))
; *                                                                   *
; *********************************************************************
                               )
                              :LOKALE_CONSTRAINTS
                               '(
; ---------------------------------------------------------------------
; Im folgenden sind die lokalen Noncodesignation-CONSTRAINTS des ersten
; Operators anzugeben:
; *********************************************************************
; *                                                                   *
     (<> ?X ?Y)
     (<> ?X ?Z)
     (<> ?Y ?Z)
     (<> ?X TISCH)
; *                                                                   *
; *********************************************************************
                         )) ... <Definition des zweiten Operators analog> )
```

Das '<>'-Symbol bei der Definition lokaler Noncodesignation-Constraints ist eigentlich redundant, da keine anderen Arten lokaler Constraints vorkommen können, wird aber dennoch vom Planer als erstes Element jeweils erwartet und muß deshalb stets angegeben werden; es dient für den Benutzer auch der besseren Lesbarkeit.

FKNLP vermeidet die Untergliederung von Effekten in add- und delete-list. Als Konvention gilt, daß nichtnegierte Nachbedingungen (z.B. (steht_auf ?X ?Y)) erzeugende und negierte Nachbedingungen (z.B. (NICHT (steht_auf ?X ?Z))) löschende Wirkung haben; das Markierungssymbol 'NICHT' wird innerhalb des Planers global definiert und muß daher in dieser Form angegeben werden.

Die Reihenfolge, in der die einzelnen Operatoren definiert werden, spielt keine Rolle. Bei der Erzeugung der Strukturen und der Liste mit den Namen aller verfügbarer Operator-Schablonen durch den Planer ergibt sich eine bestimmte Reihenfolge gemäß der Anzahl der Vorbedingungen der jeweiligen Operatoren.

Schließlich sei noch angemerkt, daß zur Unterstützung der Eingabe von Operator-Schablonen im Subdirectory "Utilities" die Datei "Operator-Rahmen" dienen kann, welche einen Operator ohne Einträge beinhaltet. (Hinweis: Eine Übersicht sämtlicher von mir angelegter Directories für diese Diplomarbeit findet sich auf der zweiten Seite des beiliegenden Quellcode-Listings.)

6.2 Programmstart

Nach Spezifikation der Problemstellung durch den Benutzer gemäß obiger Anleitung und Abspeichern derselben unter "Problemstellung.lisp" im aktuellen Directory, in dem auch Planer-Code "Planer.sbin" und Planer-Quelltext "Planer.lisp" abgespeichert sind, steht dem Progammstart nichts mehr im Wege.

Wenn durch Eingabe des Kommandos

```
cd /home/i61fs1/kienzler/Diplomarbeit
```

o.ä. sichergestellt ist, daß das Directory mit den beiden Dateien "Problemstellung.lisp" und "Planer.sbin" das aktuelle ist, muß das LISP-System durch Eingabe von

```
lisp
```

aktiviert werden, vorausgesetzt, dem System ist der Pfadname zum LISP-Interpreter bekannt. Der bereits compilierte Planer-Code wird dann auf Interpreter-Ebene mittels

```
(load "Planer.sbin")
```

geladen. Dabei ist zu beachten, daß auf den am Institut vorhandenen SUNs unterschiedliche LISP-Compiler/Interpreter installiert sind, die von demselben Quelltext verschiedene Codes generieren. "Planer.sbin" wurde auf einer SUN4 (i61S1) erzeugt, andere Maschinen compilieren zu Code-Files mit der Namenskennung ".lbin", auf denen dann "Planer.sbin" nicht ablaufbar wäre; entsprechendes Nachcompilieren ist dann notwendig. Der Planer kann natürlich auch interpretativ abgearbeitet werden, indem anstatt des Codes der Quelltext geladen wird; dies führt aber zu einer Verlangsamung um den Faktor 10-20. Schließlich wird durch Eingabe von

```
(fknlp)
```

das Programm gestartet, und der Planungsprozeß beginnt. Das Lesen der spezifizierten Problemstellung erfolgt ohne weiteres Zutun des Benutzers.

6.3 Ausgaben und Terminierung

Während des Planungsprozesses, der sich über einen größeren Zeitraum erstrecken kann, wird das Fortschreiten der Planung durch einige wenige Ausgaben am Bildschirm angezeigt. Zunächst erscheint folgender Titel:

```
  PLAN-GENERIERUNG FUER DAS GESTELLTE PROBLEM:
  -------------------------------------------

Im folgenden wird fuer jeden aktuell betrachte-
ten (unvollstaendigen) nichtlinearen Plan ange-
zeigt:
- Nummer (fortlaufend; bei 0 beginnend),
- Knoten-Anzahl (mit Start- und Zielknoten),
- Anzahl der bereits generierten, aber noch nicht
  untersuchten (unvollst.) nichtlinearen Plaene.
[bis zur eventuellen Ausgabe eines gesuchten op-
 timalen Loesungs-Plans !]
```

Es folgen zu Beginn eines jeden Planungsdurchlaufs, der die Bearbeitung eines noch unvollständigen Plans aus der Plan-Warteschlange im Hinblick auf die Erfüllung eines betrachteten Teilziels zur Aufgabe hat, Angaben zum aktuellen Plan (fortlaufende Nummer und Knoten-Anzahl einschließlich Start- und Zielknoten) und zur Länge der noch zu betrachtenden Plan-Warteschlange. Beispielsweise könnten die ersten Meldungen wie folgt lauten:

```
   Fortlaufende Nr.:   0
   Knoten-Anzahl:      2
   Plan-Warteschlange: 1

   Fortlaufende Nr.:   1
   Knoten-Anzahl:      3
   Plan-Warteschlange: 1

   Fortlaufende Nr.:   2
   Knoten-Anzahl:      4
   Plan-Warteschlange: 1

usw.

   Fortlaufende Nr.:   46
   Knoten-Anzahl:      5
   Plan-Warteschlange: 123

usw.
```

Stellt der aktuell betrachtete Plan eine Lösung dar, dann erfolgt eine vollständige, allerdings nur textuelle Ausgabe des gestellten Problems und des generierten Lösungsplans. Bezüglich eines Ausgabe-Beispiels sei auf das nächste Kapitel (Unterabschnitt 7.1.3) verwiesen, in dem die Ergebnis-Ausgabe der Lösung zum Sussman-Problem genau wiedergegeben ist.

Was die Terminierung anbelangt, sind drei mögliche Ausgänge zu unterscheiden:

- Der Planungsprozeß terminiert und gibt eine optimale Lösung für die gestellte Aufgabe aus.
- Der Planungsprozeß bricht mit der Meldung ab, daß es sich um ein unlösbares Problem handelt. Die Ausgabe lautet in diesem Fall:
  (UNLOESBARES PLANUNGSPROBLEM!)
- Der Planungsprozeß terminiert nicht.

In letzterem Fall stellt sich natürlich die Frage, ob die Terminierung noch eintritt oder nicht, aber dies ist ein unentscheidbares Problem.

6.4 Bemerkungen zu Laufzeit und Speicherbedarf

Es wurde bereits an anderer Stelle betont, daß es sich bei dem Planer auf Grund der verwendeten (erschöpfenden) Suchstrategie um ein sehr ineffizient arbeitendes Programm handelt. Dabei hängen Laufzeit und Speicherbedarf in erster Linie von der Knoten-Anzahl des zu generierenden Lösungsplans ab, falls überhaupt eine Terminierung zu verzeichnen ist.

Einige Beobachtungen bei der Lösung des Sussman-Problems mittels FKNLP (siehe auch Abschnitt 7.1) sollen dies verdeutlichen. Bis zur Ausgabe des Ergebnisses verstreicht knapp eine Minute, wenn die "abstrakteren" Operatoren (NEWTOWER und PUTON) als Schablonen zur Verfügung stehen. Bei Verwendung der "elementareren"

Operatoren (PUTDOWN, STACK, UNSTACK und PICKUP) explodieren die Laufzeit in den Stunden- und der dynamische Speicherbedarf in den Megabyte-Bereich. Nur durch zusätzliche Modifikationen auf der obersten Kontrollebene zur Einschränkung der jeweils in Betracht zu ziehenden Plan-Alternativen könne diese Größen auf ca. 10 Minuten und 2 MByte reduziert werden.

Obige Angaben sind nicht als absolut zu sehen, sondern sollen lediglich einen Eindruck der Größenverhältnisse vermitteln. Ein Faktor, der viel Zeit kostet, ist die immer wieder notwendige Speicherbereinigung (garbage collection), die auf Grund des hohen Speicherbedarfs für die generierten Pläne in der Warteschlange zyklisch wiederholt werden muß.

6.5 Portabilitäts-Betrachtungen

Ein Programm wird bekanntlich als portabel bezeichnet, wenn es mit relativ geringem Aufwand auf andere Systeme "verpflanzt" werden kann. Dieser Programmanforderung wurde bei dem implementierten Planer insofern Genüge getan, daß auf keinerlei speziellen implementationsspezifischen LISP-Funktionen oder UNIX-Betriebssystemroutinen aufgesetzt wurde, da ja auch die Ein-/Ausgabe sehr rudimentär gehalten ist. Folglich müßte der Planer ohne Anpassungsmodifikationen auf anderen LIPS-Maschinen laufen, wenn auch kein entsprechender Versuch im Rahmen dieser Arbeit unternommen worden ist.

```
================================
Kapitel 7:    Planungs-Beispiele
================================
```

Den Abschluß dieser Ausarbeitung sollen zwei Planungs-Beispiele aus der Blockwelt-Domäne bilden.

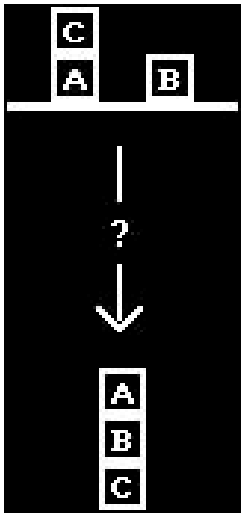
7.1 Beispiel 1: Lösung des Sussman-Problems mit FKNLP

Dieses auch in den vorigen Kapiteln viel zitierte Beispiel unterstreicht die Nützlichkeit des nichtlinearen Planungsansatzes und soll deshalb auch hier nicht fehlen.

7.1.1 Spezifikation der Problemstellung

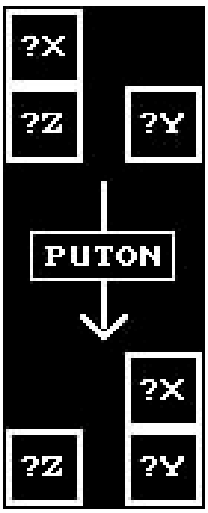
Zum Lösen der Sussman-Anomalie mittels zweier Operatoren, NEWTOWER und PUTON, ist die Problemstellung wie folgt zu spezifizieren:

```
; ============================================================================
;                          Problemspezifikation
; ============================================================================
```



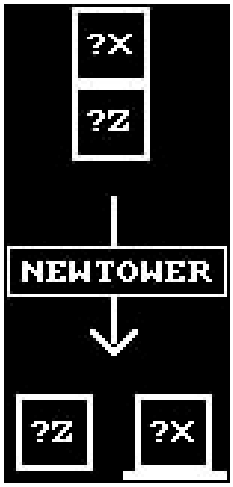

```
  (DEFUN STARTOPERATOR ()
        (MAKE-OPERATOR
             :NACHBEDINGUNGEN
                      '(

(steht_auf C A)
(steht_auf A Tisch)
(steht_auf B Tisch)
(ist_frei C)
(ist_frei B)

                       )
              :LOKALE_CONSTRAINTS
                      '(
                       ) ))
```

```
  (DEFUN ZIELOPERATOR ()
        (MAKE-OPERATOR
             :VORBEDINGUNGEN
                      '(

(steht_auf A B)
(steht_auf B C)

                       )
             :LOKALE_CONSTRAINTS
                      '(
                       ) ))

; ============================================================================
;                         Operatorspezifikation
; ============================================================================

  (DEFUN OPERATOREN ()
   '(
```




```
PUTON

      (MAKE-OPERATOR :VORBEDINGUNGEN
                      '(

(steht_auf ?X ?Z)
(ist_frei ?X)
(ist_frei ?Y)

                       )
                     :NACHBEDINGUNGEN
                      '(

(steht_auf ?X ?Y)
(ist_frei ?Z)
(NICHT (steht_auf ?X ?Z))
(NICHT (ist_frei ?Y))

                       )
                     :LOKALE_CONSTRAINTS
                      '(

(<> ?X ?Y)
(<> ?X ?Z)
```

```
(<> ?Y ?Z)
(<> ?X TISCH)

                          ) )
```




```
NEWTOWER

      (MAKE-OPERATOR :VORBEDINGUNGEN
                      '(

(steht_auf ?X ?Z)
(ist_frei ?X)

                       )
                     :NACHBEDINGUNGEN
                      '(

(steht_auf ?X TISCH)
(ist_frei ?Z)
(NICHT (steht_auf ?X ?Z))

                       )
                     :LOKALE_CONSTRAINTS
                      '(

(<> ?X ?Z)
(<> ?X TISCH)
(<> ?Z TISCH)

                       ) )
    )
  )
```

7.1.2 Planungszwischenstadien

Die während einer Aufgabenbearbeitung durch FKNLP generierten Pläne werden normalerweise nicht ausgegeben, um die erforderliche Rechenzeit nicht unnötigerweise zu vergrößern und die Anzeige am Bildschirm nicht zu unübersichtlich werden zu lassen. Erst der gefundene Lösungsplan wird angezeigt. Um einen Einblick in den Planungsprozeß zu erhalten, werden im folgenden die ersten drei generierten noch unvollständigen Pläne herausgegriffen und näher betrachtet, bevor dann in 7.1.3 die Ergebnis-Ausgabe erscheint.

Zu der gewählten Darstellung seien noch folgende Punkte erwähnt:
- Das Symbol '=' (-> Codesignation) ist als Kurzschreibweise für '=!' zu lesen, und '<>' (-> Noncodesignation) steht für '/=!'.
- Die Angabe von Bezugsknoten-Namen in eckigen Klammern direkt unterhalb von Variablen oder Merkmalen dient der genauen Kennzeichnung, auf welchen anderen Knoten im Plan sich Variable oder Merkmal beziehen.
- Unter der Rubrik "VARIABLEN" erfolgt die Darstellung aller Argument-Variablen (in Listenform). Für jede Variable werden anschließend sämtliche Merkmalargumente aufgelistet, mit denen die betreffende Variable codesigniert oder noncodesigniert (Darstellung der transitiven Hülle der Wert-Relation). Für Variablen, die mit einer Konstanten codesignieren, wird aber nur eben diese Instantiierung angezeigt, da damit die übrigen Wert-Bindungen, zumindest bei der Ausgabe der betreffenden Variablen, nicht mehr von Interesse sind.
- "OPERATOR" bezeichnet die angewandte Operator-Schablone.
- Unter der Rubrik "MERKMAL-BINDUNGEN" werden sämtliche Vor- und Nachbedingungs-Merkmale des angewandten Operators aufgelistet, auch wenn sie nicht in Wert-Relation zu anderen Merkmalen stehen. Bei Bindungen werden nur die direkt (non-)codesignierenden Merkmale angegeben.
- Unter der Rubrik "VORGAENGER-KNOTEN" sind alle direkten Vorgänger aufgeführt. Alle Knoten mit Ausnahme des Start-Knotens haben mindestens einen direkten Vorgänger.
- Unter der Rubrik "ABHAENGIGKEITEN" stehen diejenigen Vorbedingungen, die durch den jeweils an zweiter Stelle angeführten Knoten erzeugt werden.
- Unter der Rubrik "MERKMAL-ERZEUGUNGEN" stehen diejenigen Nachbedingungen, die für den jeweils an zweiter Stelle angeführten Knoten ein Vorbedingungs-Merkmal erzeugen.

Initialplan-Ausgabe:

```
Fortlaufende Nr.:    0
Knoten-Anzahl:       2
Plan-Warteschlange:  1
------------------------------------------------------------

  KNOTEN-START:
  -------------
VARIABLEN
OPERATOR             START
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN
 AN NACHBEDINGUNGEN  (STEHT_AUF C A)
                     (STEHT_AUF A TISCH)
                     (STEHT_AUF B TISCH)
                     (IST_FREI C)
                     (IST_FREI B)
VORGAENGER-KNOTEN
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN
```

```
  KNOTEN-ZIEL:
  ------------
VARIABLEN
OPERATOR              ZIEL
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF A B)
                      (STEHT_AUF B C)
 AN NACHBEDINGUNGEN
VORGAENGER-KNOTEN     KNOTEN-START
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN

-------------------------------------------------------------
```

Nach dieser Ausgabe des Initialplans soll sich der als nächster von FKNLP generierte Plan gleich anschließen, in den bereits ein weiterer Operator eingebaut ist.

(Anm.: Die Namen der Knoten ergeben sich aus dem Einbauzeitpunkt in den Plan, nicht aus der Anwendungsreihenfolge!)

Ausgabe des zweiten erzeugten Plans:

```
Fortlaufende Nr.:    1
Knoten-Anzahl:       3
Plan-Warteschlange:  1
-------------------------------------------------------------

  KNOTEN-START:
  -------------
VARIABLEN
OPERATOR              START
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN
 AN NACHBEDINGUNGEN   (STEHT_AUF C A)
                      (STEHT_AUF A TISCH)
                      (STEHT_AUF B TISCH)
                      (IST_FREI C)
                      (IST_FREI B)
VORGAENGER-KNOTEN
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN

  KNOTEN-1:
  ---------
VARIABLEN             (?Y ?X ?Z)
                      ?Y =  B
                      ?X =  A
                      ?Z <> B
                            A
                            ?Y
                             [KNOTEN-1]
                            ?X
                             [KNOTEN-1]
OPERATOR              PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF ?X ?Z)
                      (IST_FREI ?X)
                      (IST_FREI ?Y)
 AN NACHBEDINGUNGEN   (STEHT_AUF ?X ?Y) =  (STEHT_AUF A B)
                                            [KNOTEN-ZIEL]
```

```
                    (IST_FREI ?Z)
                    (NICHT (STEHT_AUF ?X ?Z))
                    (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN   KNOTEN-START
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-ZIEL:
  ------------
VARIABLEN
OPERATOR            ZIEL
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN  (STEHT_AUF A B) =  (STEHT_AUF ?X ?Y)
                                        [KNOTEN-1]
                    (STEHT_AUF B C)
 AN NACHBEDINGUNGEN
VORGAENGER-KNOTEN   KNOTEN-1
ABHAENGIGKEITEN     (STEHT_AUF A B)  VON  KNOTEN-1
MERKMAL-ERZEUGUNGEN
```

Erklärung für die Generierung des obigen Plans:
Als das erste zu betrachtende Teilziel wird die unerfüllte Vorbedingung (steht_auf A B) des Zielknotens bestimmt. Zu deren Erzeugung kann im bestehenden (Initial-)Plan kein geeigneter Knoten gefunden werden, weswegen es des Einbaus eines neuen Operators, PUTON, bedarf, der über die mit dem betrachteten Teilziel codesignierbare erzeugende Nachbedingung (steht_auf ?X ?Y) verfügt. Durch Setzen eines Constraints zwischen Erzeuger-Effekt und Teilziel erfolgt die Unifikation oder Codesignation beider Merkmale. Diese wirkt sich auf die Argument-Variablen von Knoten-1 durch entsprechende Instantiierungen von ?X (= A) und ?Y (= B) aus. Die dritte Variable, ?Z, noncodesigniert mit einer Reihe von Konstanten und Variablen anderer Knoten, was in der Existenz lokaler Constraints bei PUTON begründet liegt.

Schließlich soll der von FKNLP eingeschlagene Weg auf der Suche nach der Lösung noch ein Plan weiterverfolgt werden.

Ausgabe des dritten erzeugten Plans:

```
Fortlaufende Nr.:    2
Knoten-Anzahl:       4
Plan-Warteschlange:  1
```

```
  KNOTEN-START:
  -------------
VARIABLEN
OPERATOR            START
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN
 AN NACHBEDINGUNGEN (STEHT_AUF C A)
                    (STEHT_AUF A TISCH)
                    (STEHT_AUF B TISCH)
                    (IST_FREI C)
                    (IST_FREI B)
VORGAENGER-KNOTEN
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN
```

```
  KNOTEN-1:
  ---------
VARIABLEN            (?Y ?X ?Z)
                     ?Y =  B
                     ?X =  A
                     ?Z <> B
                           A
                           ?Y
                            [KNOTEN-1]
                           ?X
                            [KNOTEN-1]
                           ?X
                            [KNOTEN-2]
OPERATOR             PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN   (STEHT_AUF ?X ?Z)
                     (IST_FREI ?X)
                     (IST_FREI ?Y)
 AN NACHBEDINGUNGEN  (STEHT_AUF ?X ?Y) =  (STEHT_AUF A B)
                                           [KNOTEN-ZIEL]
                     (IST_FREI ?Z)
                     (NICHT (STEHT_AUF ?X ?Z))
                     (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN    KNOTEN-START
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN  (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-2:
  ---------
VARIABLEN            (?Y ?X ?Z)
                     ?Y =  C
                     ?X =  B
                     ?Z <> C
                           B
                           ?Y
                            [KNOTEN-2]
                           ?X
                            [KNOTEN-2]
                           ?Y
                            [KNOTEN-1]
OPERATOR             PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN   (STEHT_AUF ?X ?Z)
                     (IST_FREI ?X)
                     (IST_FREI ?Y)
 AN NACHBEDINGUNGEN  (STEHT_AUF ?X ?Y) =  (STEHT_AUF B C)
                                           [KNOTEN-ZIEL]
                     (IST_FREI ?Z)
                     (NICHT (STEHT_AUF ?X ?Z))
                     (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN    KNOTEN-START
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN  (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-ZIEL:
  ------------
VARIABLEN
OPERATOR             ZIEL
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN   (STEHT_AUF A B) =  (STEHT_AUF ?X ?Y)
                                         [KNOTEN-1]
                     (STEHT_AUF B C) =  (STEHT_AUF ?X ?Y)
                                         [KNOTEN-2]
```

```
 AN NACHBEDINGUNGEN
VORGAENGER-KNOTEN    KNOTEN-2
                     KNOTEN-1
ABHAENGIGKEITEN      (STEHT_AUF A B)  VON  KNOTEN-1
                     (STEHT_AUF B C)  VON  KNOTEN-2
MERKMAL-ERZEUGUNGEN

------------------------------------------------------------
```

Erklärung für die Generierung des obigen dritten Plans:
Als nächstes Teilziel wird die zweite unerfüllte Vorbedingung (steht_auf B C) des Zielknotens betrachtet. Die erzeugende Nachbedingung (steht_auf ?X ?Y) des PUTON-Knotens ist auf Grund bestehender Wert-Bindungen mit dem aktuellen Teilziel nicht codesignierbar. Daher scheidet KNOTEN-1 als Erzeuger aus. Da auch keine anderen geeigneten Operator-Instanzen im Plan auszumachen sind, wird wiederum der Einbau eines neuen PUTON-Knotens notwendig, der den Namen KNOTEN-2 erhält. Ein Constraint sorgt für die Codesignation von Erzeuger-Effekt und Teilziel. KNOTEN-1 und KNOTEN-2 sind direkte Vorgänger zum Zielknoten. Auf Grund dieser bestehenden partiellen Operator-Ordnung könnten sich die beiden PUTON-Knoten gegenseitig bezüglich der Erzeugungen für den Zielknoten stören, was aber wegen der bestehenden Wert-Bindungen nicht der Fall ist.

FKNLP erzeugt nun weitere 167 Pläne, bevor es die optimale Lösung findet. Die Vorgehensweise ist dabei allerdings nicht immer so zielstrebig, wie dies die ersten generierten Pläne vielleicht vermuten lassen, denn bei diesen kamen noch keine unterschiedlichen Erzeuger in Frage, und es bestanden auch keine Konflikte, die aufgelöst werden mußten. Von den insgesamt 170 generierten Plänen werden 47 näher betrachtet; darunter sind es wiederum letztlich nur 11 Pläne (einschließlich Initial- und Ergebnisplan), die das sukzessive Fortschreiten der Planung bis hin zur optimalen Lösung beschreiben.

7.1.3 Ergebnis-Ausgabe

Der Lösungsplan besteht aus fünf Knoten (einschließlich Start- und Zielknoten) und zehn Wert-Constraints. Die bestehende Ordnung ist in diesem Fall sogar total; es liegt also ein linearer Plan vor, was durch den nichtlinearen Planungsansatz ja nicht ausgeschlossen ist. Sämtliche im Lösungsplan enthaltenen Variablen sind instantiiert.

Ausgabe des Lösungsplans:

```
Fortlaufende Nr.:    46
Knoten-Anzahl:        5
Plan-Warteschlange: 123

------------------------------------------------------------
GESTELLTES PLANUNGSPROBLEM:
------------------------------------------------------------

GEGEBENE STARTSITUATION:
========================
   (STEHT_AUF C A)
   (STEHT_AUF A TISCH)
   (STEHT_AUF B TISCH)
   (IST_FREI C)
   (IST_FREI B)
```

```
ERWUENSCHTE ZIELSITUATION:
==========================
   (STEHT_AUF A B)
   (STEHT_AUF B C)

------------------------------------------------------------
NICHTLINEARER PLAN ZUR LOESUNG DES PROBLEMS:
------------------------------------------------------------

  KNOTEN-START:
  -------------
VARIABLEN
OPERATOR              START
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN
 AN NACHBEDINGUNGEN   (STEHT_AUF C A) =  (STEHT_AUF ?X ?Z)
                                          [KNOTEN-3]
                      (STEHT_AUF A TISCH) =  (STEHT_AUF ?X ?Z)
                                              [KNOTEN-1]
                      (STEHT_AUF B TISCH) =  (STEHT_AUF ?X ?Z)
                                              [KNOTEN-2]
                      (IST_FREI C) =  (IST_FREI ?X)
                                       [KNOTEN-3]
                                      (IST_FREI ?Y)
                                       [KNOTEN-2]
                      (IST_FREI B) =  (IST_FREI ?Y)
                                       [KNOTEN-1]
                                      (IST_FREI ?X)
                                       [KNOTEN-2]
VORGAENGER-KNOTEN
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN   (IST_FREI C)  FUER  KNOTEN-3
                      (STEHT_AUF C A)  FUER  KNOTEN-3
                      (STEHT_AUF B TISCH)  FUER  KNOTEN-2
                      (IST_FREI C)  FUER  KNOTEN-2
                      (IST_FREI B)  FUER  KNOTEN-2
                      (STEHT_AUF A TISCH)  FUER  KNOTEN-1
                      (IST_FREI B)  FUER  KNOTEN-1

  KNOTEN-3:
  ---------
VARIABLEN             (?X ?Z)
                      ?X =  C
                      ?Z =  A
OPERATOR              NEWTOWER
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF ?X ?Z) =  (STEHT_AUF C A)
                                            [KNOTEN-START]
                      (IST_FREI ?X) =  (IST_FREI C)
                                        [KNOTEN-START]
 AN NACHBEDINGUNGEN   (STEHT_AUF ?X TISCH)
                      (IST_FREI ?Z) =  (IST_FREI ?X)
                                        [KNOTEN-1]
                      (NICHT (STEHT_AUF ?X ?Z))
VORGAENGER-KNOTEN     KNOTEN-START
ABHAENGIGKEITEN       (STEHT_AUF ?X ?Z)  VON  KNOTEN-START
                      (IST_FREI ?X)  VON  KNOTEN-START
MERKMAL-ERZEUGUNGEN   (IST_FREI ?Z)  FUER  KNOTEN-1

  KNOTEN-2:
  ---------
VARIABLEN             (?Y ?X ?Z)
```

```
                    ?Y =  C
                    ?X =  B
                    ?Z =  TISCH
OPERATOR            PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN  (STEHT_AUF ?X ?Z) =  (STEHT_AUF B TISCH)
                                          [KNOTEN-START]
                    (IST_FREI ?X) =  (IST_FREI B)
                                      [KNOTEN-START]
                    (IST_FREI ?Y) =  (IST_FREI C)
                                      [KNOTEN-START]
 AN NACHBEDINGUNGEN (STEHT_AUF ?X ?Y) =  (STEHT_AUF B C)
                                          [KNOTEN-ZIEL]
                    (IST_FREI ?Z)
                    (NICHT (STEHT_AUF ?X ?Z))
                    (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN   KNOTEN-3
ABHAENGIGKEITEN     (IST_FREI ?X)  VON  KNOTEN-START
                    (IST_FREI ?Y)  VON  KNOTEN-START
                    (STEHT_AUF ?X ?Z)  VON  KNOTEN-START
MERKMAL-ERZEUGUNGEN (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-1:
  ---------
VARIABLEN           (?Y ?X ?Z)
                    ?Y =  B
                    ?X =  A
                    ?Z =  TISCH
OPERATOR            PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN  (STEHT_AUF ?X ?Z) =  (STEHT_AUF A TISCH)
                                          [KNOTEN-START]
                    (IST_FREI ?X) =  (IST_FREI ?Z)
                                      [KNOTEN-3]
                    (IST_FREI ?Y) =  (IST_FREI B)
                                      [KNOTEN-START]
 AN NACHBEDINGUNGEN (STEHT_AUF ?X ?Y) =  (STEHT_AUF A B)
                                          [KNOTEN-ZIEL]
                    (IST_FREI ?Z)
                    (NICHT (STEHT_AUF ?X ?Z))
                    (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN   KNOTEN-2
ABHAENGIGKEITEN     (IST_FREI ?X)  VON  KNOTEN-3
                    (IST_FREI ?Y)  VON  KNOTEN-START
                    (STEHT_AUF ?X ?Z)  VON  KNOTEN-START
MERKMAL-ERZEUGUNGEN (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-ZIEL:
  ------------
VARIABLEN
OPERATOR            ZIEL
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN  (STEHT_AUF A B) =  (STEHT_AUF ?X ?Y)
                                        [KNOTEN-1]
                    (STEHT_AUF B C) =  (STEHT_AUF ?X ?Y)
                                        [KNOTEN-2]
 AN NACHBEDINGUNGEN
VORGAENGER-KNOTEN   KNOTEN-1
ABHAENGIGKEITEN     (STEHT_AUF A B)  VON  KNOTEN-1
                    (STEHT_AUF B C)  VON  KNOTEN-2
MERKMAL-ERZEUGUNGEN

------------------------------------------------------------
```

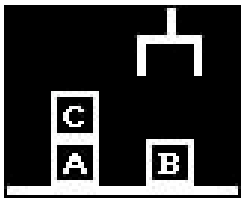
Die lösende Operator-Sequenz lautet also:

1. Aktion: NEWTOWER (C, A),
2. Aktion: PUTON (B, C, TISCH),
3. Aktion: PUTON (A, B, TISCH).

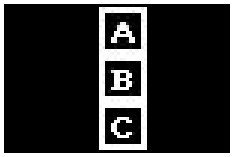
7.1.4 Weitere Lösung mit modifizierten Operatoren

Die Sussman-Problemstellung kann in Anlehnung an [Hertzberg 1989] dahingehend abgewandelt werden, daß die zur Verfügung stehenden Operatoren von elementarerer Wirkung sind als NEWTOWER und PUTON, die den Zustand einer möglichen Greiferhand zur Manipulation der Klötze nicht berücksichtigen. Start- und Zielsituation und die Operator-Schablonen PICKUP, PUTDOWN, STACK UND UNSTACK werden dann wie folgt definiert:

```
; ============================================================================
;                          Problemspezifikation
; ============================================================================

  (DEFUN STARTOPERATOR ()
        (MAKE-OPERATOR
             :NACHBEDINGUNGEN
                      '(
```

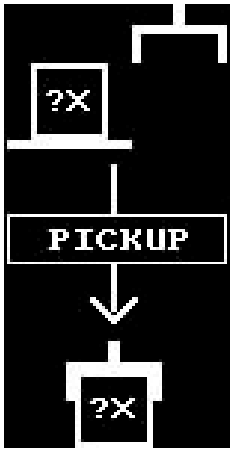



```
(steht_auf C A)
(steht_auf_Tisch A)
(steht_auf_Tisch B)
(ist_frei C)
(ist_frei B)
(Hand_haelt NIL)

                       )
              :LOKALE_CONSTRAINTS
                      '(
                       ) ))

  (DEFUN ZIELOPERATOR ()
        (MAKE-OPERATOR
             :VORBEDINGUNGEN
                      '(
```



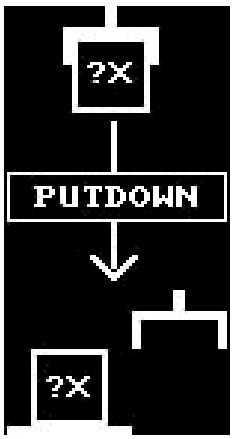

```
(steht_auf A B)
(steht_auf B C)
```

```
                        )
              :LOKALE_CONSTRAINTS
                       '(
                        ) ))


; ============================================================================
;                         Operatorspezifikation
; ============================================================================

  (DEFUN OPERATOREN ()
   '(
```



PICKUP

```
      (MAKE-OPERATOR :VORBEDINGUNGEN
                      '(

(steht_auf_Tisch ?X)
(ist_frei ?X)
(Hand_haelt NIL)

                       )
                     :NACHBEDINGUNGEN
                      '(

(Hand_haelt ?X)
(NICHT (steht_auf_Tisch ?X))
(NICHT (ist_frei ?X))
(NICHT (Hand_haelt NIL))

                       )
                     :LOKALE_CONSTRAINTS
                      '(
                       ) )
```



PUTDOWN

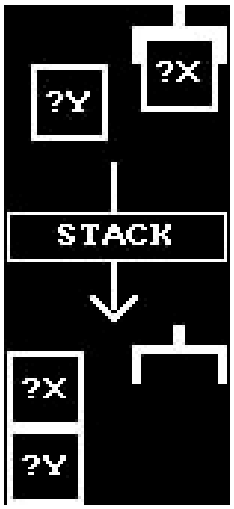
```
      (MAKE-OPERATOR :VORBEDINGUNGEN
                     '(
```

(Hand_haelt ?X)

```
                      )
                     :NACHBEDINGUNGEN
                      '(
```

(steht_auf_Tisch ?X)
(ist_frei ?X)
(Hand_haelt NIL)
(NICHT (Hand_haelt ?X))

```
                      )
                     :LOKALE_CONSTRAINTS
                      '(
                       ) )
```

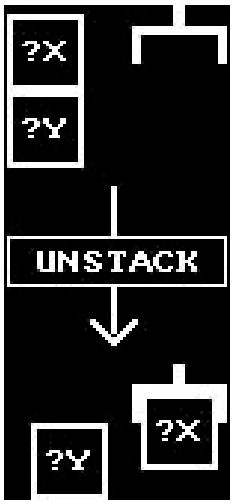


STACK

```
      (MAKE-OPERATOR :VORBEDINGUNGEN
                     '(
```

(ist_frei ?Y)
(Hand_haelt ?X)

```
                        )
                       :NACHBEDINGUNGEN
                        '(

(Hand_haelt NIL)
(steht_auf ?X ?Y)
(ist_frei ?X)
(NICHT (ist_frei ?Y))
(NICHT (Hand_haelt ?X))

                        )
                       :LOKALE_CONSTRAINTS
                        '(

(<> ?X ?Y)

                        ) )
```




```
UNSTACK

      (MAKE-OPERATOR :VORBEDINGUNGEN
                      '(

(Hand_haelt NIL)
(ist_frei ?X)
(steht_auf ?X ?Y)

                        )
                       :NACHBEDINGUNGEN
                        '(

(Hand_haelt ?X)
(ist_frei ?Y)
(NICHT (Hand_haelt NIL))
(NICHT (ist_frei ?X))
(NICHT (steht_auf ?X ?Y))

                        )
                       :LOKALE_CONSTRAINTS
                        '(

(<> ?X ?Y)

                        )))
  )
```

Damit liefert FKNLP den auf den folgenden Seiten abgebildeten und aus acht Knoten und 17 Wert-Constraints bestehenden linearen Lösungsplan, dessen Variablen allesamt an Konstanten gebunden sind.

```
------------------------------------------------------------
GESTELLTES PLANUNGSPROBLEM:
------------------------------------------------------------

GEGEBENE STARTSITUATION:
========================
   (STEHT_AUF C A)
   (STEHT_AUF_TISCH A)
   (STEHT_AUF_TISCH B)
   (IST_FREI C)
   (IST_FREI B)
   (HAND_HAELT NIL)

ERWUENSCHTE ZIELSITUATION:
==========================
   (STEHT_AUF A B)
   (STEHT_AUF B C)

------------------------------------------------------------
NICHTLINEARER PLAN ZUR LOESUNG DES PROBLEMS:
------------------------------------------------------------

  KNOTEN-START:
  -------------
VARIABLEN
OPERATOR              START
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN
 AN NACHBEDINGUNGEN   (STEHT_AUF C A) =  (STEHT_AUF ?X ?Y)
                                          [KNOTEN-5]
                      (STEHT_AUF_TISCH A) =  (STEHT_AUF_TISCH ?X)
                                              [KNOTEN-3]
                      (STEHT_AUF_TISCH B) =  (STEHT_AUF_TISCH ?X)
                                              [KNOTEN-4]
                      (IST_FREI C) =  (IST_FREI ?X)
                                       [KNOTEN-5]
                                      (IST_FREI ?Y)
                                       [KNOTEN-2]
                      (IST_FREI B) =  (IST_FREI ?X)
                                       [KNOTEN-4]
                      (HAND_HAELT NIL) =  (HAND_HAELT NIL)
                                           [KNOTEN-5]
                                          (HAND_HAELT NIL)
                                           [KNOTEN-4]
VORGAENGER-KNOTEN
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN   (STEHT_AUF C A)  FUER  KNOTEN-5
                      (IST_FREI C)  FUER  KNOTEN-5
                      (HAND_HAELT NIL)  FUER  KNOTEN-5
                      (HAND_HAELT NIL)  FUER  KNOTEN-4
                      (IST_FREI B)  FUER  KNOTEN-4
                      (STEHT_AUF_TISCH B)  FUER  KNOTEN-4
                      (STEHT_AUF_TISCH A)  FUER  KNOTEN-3
                      (IST_FREI C)  FUER  KNOTEN-2

  KNOTEN-5:
  ---------
VARIABLEN             (?Y ?X)
```

```
                    ?Y =  A
                    ?X =  C
OPERATOR            UNSTACK
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN  (HAND_HAELT NIL) =  (HAND_HAELT NIL)
                                         [KNOTEN-START]
                    (IST_FREI ?X) =  (IST_FREI C)
                                      [KNOTEN-START]
                    (STEHT_AUF ?X ?Y) =  (STEHT_AUF C A)
                                          [KNOTEN-START]

 AN NACHBEDINGUNGEN (HAND_HAELT ?X) =  (HAND_HAELT ?X)
                                        [KNOTEN-6]
                    (IST_FREI ?Y) =  (IST_FREI ?X)
                                      [KNOTEN-3]
                    (NICHT (HAND_HAELT NIL)) =  (HAND_HAELT NIL)
                                                 [KNOTEN-6]
                    (NICHT (IST_FREI ?X)) =  (IST_FREI ?X)
                                              [KNOTEN-6]
                    (NICHT (STEHT_AUF ?X ?Y))
VORGAENGER-KNOTEN   KNOTEN-START
ABHAENGIGKEITEN     (HAND_HAELT NIL)  VON  KNOTEN-START
                    (IST_FREI ?X)  VON  KNOTEN-START
                    (STEHT_AUF ?X ?Y)  VON  KNOTEN-START
MERKMAL-ERZEUGUNGEN (HAND_HAELT ?X)  FUER  KNOTEN-6
                    (IST_FREI ?Y)  FUER  KNOTEN-3

  KNOTEN-6:
  ---------
VARIABLEN           (?X)
                    ?X =  C
OPERATOR            PUTDOWN
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN  (HAND_HAELT ?X) =  (HAND_HAELT ?X)
                                        [KNOTEN-5]
 AN NACHBEDINGUNGEN (STEHT_AUF_TISCH ?X)
                    (IST_FREI ?X) =  (IST_FREI ?X)
                                      [KNOTEN-5]
                    (HAND_HAELT NIL) =  (HAND_HAELT NIL)
                                         [KNOTEN-5]
                    (NICHT (HAND_HAELT ?X))
VORGAENGER-KNOTEN   KNOTEN-5
ABHAENGIGKEITEN     (HAND_HAELT ?X)  VON  KNOTEN-5
MERKMAL-ERZEUGUNGEN (HAND_HAELT NIL)  FUER  KNOTEN-4
                    (IST_FREI ?X)  FUER  KNOTEN-2

  KNOTEN-4:
  ---------
VARIABLEN           (?X)
                    ?X =  B
OPERATOR            PICKUP
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN  (STEHT_AUF_TISCH ?X) =  (STEHT_AUF_TISCH B)
                                             [KNOTEN-START]
                    (IST_FREI ?X) =  (IST_FREI ?B)
                                      [KNOTEN-START]
                    (HAND_HAELT NIL) =  (HAND_HAELT NIL)
                                         [KNOTEN-START]
 AN NACHBEDINGUNGEN (HAND_HAELT ?X) =  (HAND_HAELT ?X)
                                        [KNOTEN-2]
                    (NICHT (STEHT_AUF_TISCH ?X))
                    (NICHT (IST_FREI ?X))
                    (NICHT (HAND_HAELT NIL))
```

```
VORGAENGER-KNOTEN     KNOTEN-6
ABHAENGIGKEITEN       (STEHT_AUF_TISCH ?X)  VON  KNOTEN-START
                      (IST_FREI ?X)  VON  KNOTEN-START
                      (HAND_HAELT NIL)  VON  KNOTEN-START
MERKMAL-ERZEUGUNGEN   (HAND_HAELT ?X)  FUER  KNOTEN-2

  KNOTEN-2:
  ---------
VARIABLEN             (?X ?Y)
                      ?X =  B
                      ?Y =  C
OPERATOR              STACK
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (IST_FREI ?Y) =  (IST_FREI C)
                                        [KNOTEN-START]

                      (HAND_HAELT ?X) =  (HAND_HAELT ?X)
                                          [KNOTEN-4]

 AN NACHBEDINGUNGEN   (HAND_HAELT NIL) =  (HAND_HAELT NIL)
                                           [KNOTEN-3]
                      (STEHT_AUF ?X ?Y) =  (STEHT_AUF B C)
                                            [KNOTEN-ZIEL]
                      (IST_FREI ?X) =  (IST_FREI ?Y)
                                        [KNOTEN-1]
                      (NICHT (IST_FREI ?Y))
                      (NICHT (HAND_HAELT ?X))
VORGAENGER-KNOTEN     KNOTEN-4
ABHAENGIGKEITEN       (IST_FREI ?Y)  VON  KNOTEN-START
                      (HAND_HAELT ?X)  VON  KNOTEN-4
MERKMAL-ERZEUGUNGEN   (HAND_HAELT NIL)  FUER  KNOTEN-3
                      (IST_FREI ?X)  FUER  KNOTEN-1
                      (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-3:
  ---------
VARIABLEN             (?X)
                      ?X =  A
OPERATOR              PICKUP
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF_TISCH ?X) =  (STEHT_AUF_TISCH A)
                                               [KNOTEN-START]
                      (IST_FREI ?X) =  (IST_FREI ?Y)
                                        [KNOTEN-5]
                      (HAND_HAELT NIL) =  (HAND_HAELT NIL)
                                           [KNOTEN-2]
 AN NACHBEDINGUNGEN   (HAND_HAELT ?X) =  (HAND_HAELT ?X)
                                          [KNOTEN-1]
                      (NICHT (STEHT_AUF_TISCH ?X))
                      (NICHT (IST_FREI ?X))
                      (NICHT (HAND_HAELT NIL))
VORGAENGER-KNOTEN     KNOTEN-2
ABHAENGIGKEITEN       (STEHT_AUF_TISCH ?X)  VON  KNOTEN-START
                      (IST_FREI ?X)  VON  KNOTEN-5
                      (HAND_HAELT NIL)  VON  KNOTEN-2
MERKMAL-ERZEUGUNGEN   (HAND_HAELT ?X)  FUER  KNOTEN-1

  KNOTEN-1:
  ---------
VARIABLEN             (?X ?Y)
                      ?X =  A
                      ?Y =  B
OPERATOR              STACK
```

```
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN   (IST_FREI ?Y) =  (IST_FREI ?X)
                                       [KNOTEN-2]
                     (HAND_HAELT ?X) =  (HAND_HAELT ?X)
                                         [KNOTEN-3]
 AN NACHBEDINGUNGEN  (HAND_HAELT NIL)
                     (STEHT_AUF ?X ?Y) =  (STEHT_AUF A B)
                                           [KNOTEN-ZIEL]
                     (IST_FREI ?X)
                     (NICHT (IST_FREI ?Y))
                     (NICHT (HAND_HAELT ?X))
VORGAENGER-KNOTEN    KNOTEN-3
ABHAENGIGKEITEN      (HAND_HAELT ?X)  VON  KNOTEN-3
                     (IST_FREI ?Y)  VON  KNOTEN-2
MERKMAL-ERZEUGUNGEN  (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-ZIEL:
  ------------
VARIABLEN
OPERATOR             ZIEL
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN   (STEHT_AUF A B) =  (STEHT_AUF ?X ?Y)
                                         [KNOTEN-1]
                     (STEHT_AUF B C) =  (STEHT_AUF ?X ?Y)
                                         [KNOTEN-2]
 AN NACHBEDINGUNGEN
VORGAENGER-KNOTEN    KNOTEN-1
ABHAENGIGKEITEN      (STEHT_AUF A B)  VON  KNOTEN-1
                     (STEHT_AUF B C)  VON  KNOTEN-2
MERKMAL-ERZEUGUNGEN
```

Die lösende Operator-Sequenz lautet also:

1. Aktion: UNSTACK (C, A),
2. Aktion: PUTDOWN (C),
3. Aktion: PICKUP (B),
4. Aktion: STACK (B, C),
5. Aktion: PICKUP (A),
6. Aktion: STACK (A, B).

Bemerkungen zu obigem Plan:

- An zwei Stellen (genauer: bei zwei Nachbedingungen von KNOTEN-5) tauchen Codesignations-Wert-Relationen zwischen einem nichtnegierten und einem negierten Merkmal auf, was definitionsgemäß nicht möglich ist:

```
   (NICHT (HAND_HAELT NIL)) =  (HAND_HAELT NIL)
                                [KNOTEN-6]
   (NICHT (IST_FREI ?X)) =  (IST_FREI ?X)
                             [KNOTEN-6]
```

  Dies ist kein Widerspruch, denn gemeint ist jeweils eine Unifikation zwischen dem nichtnegierten Part beider Merkmale. Das 'NICHT' (auf der Zerstörerseite) erscheint nur deshalb, um die löschende Wirkung der betreffenden Nachbedingung anzuzeigen. Zu einer derartigen Bindung zwischen einem löschenden (Zerstörer-) Effekt und einem erzeugenden (Retter-) Effekt kommt es nur bei Auflösung bestehender Konflikte durch Einschieben eines Retters.

- Die Sonderstellung des Arguments NIL wurde bereits an anderer Stelle betont und kann hier am Beispiel beobachtet werden: NIL wird weder als Konstante noch als Variable behandelt. Nimmt etwa das Merkmal (HAND_HAELT NIL) die Rolle des

betrachteten Teilziels ein, so kann eine Unifikation mit einem anderen erzeugenden Merkmal nur dann stattfinden, wenn es ebenfalls die Form (HAND_HAELT NIL) aufweist, also identisch ist; NIL hat hierbei dieselbe Behandlung wie eine Konstante zur Folge. Eine Unifikation mit einem Merkmal der Form (HAND_HAELT ?X), wobei die Bezeichnung der Variablen beliebig ist, würde allerdings scheitern, da bei FKNLP konventionsgemäß Instantiierung einer Variablen mit NIL ausgeschlossen wird.

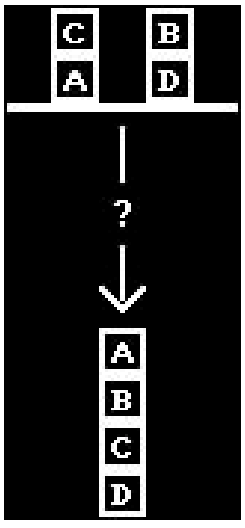
Damit dürfte das Sussman-Problem ausreichend abgehandelt sein. Ein weiteres kleines Planungs-Beispiel aus der Blockwelt sei nun noch im nachfolgenden Abschnitt betrachtet.

7.2 Beispiel 2: Lösung eines weiteren Problems aus der Blockwelt mit FKNLP

Wenigstens ein weiteres, wenn auch sehr kleines Problem aus der Blockwelt, das von der Sussman-Anomalie verschieden ist, sei noch im folgenden mit anschließender wiederum durch FKNLP generierter Lösung angeführt. Als Operatoren stehen NEWTOWER und PUTON (siehe 7.1.1) zur Verfügung.

7.2.1 Spezifikation der Problemstellung

```
; ================================================================================
;                            Problemspezifikation
; ================================================================================

  (DEFUN STARTOPERATOR ()
        (MAKE-OPERATOR
             :NACHBEDINGUNGEN
                      '(
```




```
(steht_auf C A)
(steht_auf B D)
(steht_auf A Tisch)
(steht_auf D Tisch)
(ist_frei C)
(ist_frei B)

                       )
```

```
              :LOKALE_CONSTRAINTS
                       '(
                        ) ))

  (DEFUN ZIELOPERATOR ()
        (MAKE-OPERATOR
             :VORBEDINGUNGEN
                       '(

(steht_auf A B)
(steht_auf B C)
(steht_auf C D)

                        )
             :LOKALE_CONSTRAINTS
                       '(
                        ) ))
```

7.2.2 Lösungsplan

```
Fortlaufende Nr.:    461
Knoten-Anzahl:       6
Plan-Warteschlange:  1525
------------------------------------------------------------
GESTELLTES PLANUNGSPROBLEM:
------------------------------------------------------------

GEGEBENE STARTSITUATION:
========================
   (STEHT_AUF C A)
   (STEHT_AUF B D)
   (STEHT_AUF A TISCH)
   (STEHT_AUF D TISCH)
   (IST_FREI C)
   (IST_FREI B)

ERWUENSCHTE ZIELSITUATION:
==========================
   (STEHT_AUF A B)
   (STEHT_AUF B C)
   (STEHT_AUF C D)

------------------------------------------------------------
NICHTLINEARER PLAN ZUR LOESUNG DES PROBLEMS:
------------------------------------------------------------

  KNOTEN-START:
  -------------
VARIABLEN
OPERATOR             START
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN
 AN NACHBEDINGUNGEN  (STEHT_AUF C A) =  (STEHT_AUF ?X ?Z)
                                         [KNOTEN-3]
                     (STEHT_AUF B D) =  (STEHT_AUF ?X ?Z)
                                         [KNOTEN-4]
                     (STEHT_AUF A TISCH) =  (STEHT_AUF ?X ?Z)
                                             [KNOTEN-1]
                     (STEHT_AUF D TISCH)
                     (IST_FREI C) =  (IST_FREI ?Y)
```

```
                                    [KNOTEN-2]
                                  (IST_FREI ?X)
                                    [KNOTEN-3]
                      (IST_FREI B) =(IST_FREI ?X)
                                    [KNOTEN-4]
                                  (IST_FREI ?X)
                                    [KNOTEN-2]
                                  (IST_FREI ?Y)
                                    [KNOTEN-1]
VORGAENGER-KNOTEN
ABHAENGIGKEITEN
MERKMAL-ERZEUGUNGEN   (IST_FREI B)  FUER  KNOTEN-4
                      (STEHT_AUF B D)  FUER  KNOTEN-4
                      (IST_FREI C)  FUER  KNOTEN-3
                      (STEHT_AUF C A)  FUER  KNOTEN-3
                      (IST_FREI C)  FUER  KNOTEN-2
                      (IST_FREI B)  FUER  KNOTEN-2
                      (STEHT_AUF A TISCH)  FUER  KNOTEN-1
                      (IST_FREI B)  FUER  KNOTEN-1

  KNOTEN-4:
  ---------
VARIABLEN             (?X ?Z)
                      ?X =  B
                      ?Z =  D
OPERATOR              NEWTOWER
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF ?X ?Z) =  (STEHT_AUF B D)
                                            [KNOTEN-START]
                      (IST_FREI ?X) =  (IST_FREI B)
                                        [KNOTEN-START]
 AN NACHBEDINGUNGEN   (STEHT_AUF ?X TISCH) =  (STEHT_AUF ?X ?Z)
                                               [KNOTEN-2]
                      (IST_FREI ?Z) =  (IST_FREI ?Y)
                                        [KNOTEN-3]
                      (NICHT (STEHT_AUF ?X ?Z))
VORGAENGER-KNOTEN     KNOTEN-START
ABHAENGIGKEITEN       (STEHT_AUF ?X ?Z)  VON  KNOTEN-START
                      (IST_FREI ?X)  VON  KNOTEN-START
MERKMAL-ERZEUGUNGEN   (IST_FREI ?Z)  FUER  KNOTEN-3
                      (STEHT_AUF ?X TISCH)  FUER  KNOTEN-2

  KNOTEN-3:
  ---------
VARIABLEN             (?Y ?X ?Z)
                      ?Y =  D
                      ?X =  C
                      ?Z =  A
OPERATOR              PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF ?X ?Z) =  (STEHT_AUF C A)
                                            [KNOTEN-START]
                      (IST_FREI ?X) =  (IST_FREI C)
                                        [KNOTEN-START]
                      (IST_FREI ?Y) =  (IST_FREI ?Z)
                                        [KNOTEN-4]
 AN NACHBEDINGUNGEN   (STEHT_AUF ?X ?Y) =  (STEHT_AUF C D)
                                            [KNOTEN-ZIEL]
                      (IST_FREI ?Z) =  (IST_FREI ?X)
                                        [KNOTEN-1]
                      (NICHT (STEHT_AUF ?X ?Z))
                      (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN     KNOTEN-4
```

```
ABHAENGIGKEITEN       (STEHT_AUF ?X ?Z)  VON  KNOTEN-START
                      (IST_FREI ?X)  VON  KNOTEN-START
                      (IST_FREI ?Y)  VON  KNOTEN-4
MERKMAL-ERZEUGUNGEN   (IST_FREI ?Z)  FUER  KNOTEN-1
                      (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-2:
  ---------
VARIABLEN             (?Y ?X ?Z)
                      ?Y =  C
                      ?X =  B
                      ?Z =  TISCH
OPERATOR              PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF ?X ?Z) =  (STEHT_AUF ?X TISCH)
                                           [KNOTEN-4]
                      (IST_FREI ?X) =  (IST_FREI B)
                                        [KNOTEN-START]
                      (IST_FREI ?Y) =  (IST_FREI C)
                                        [KNOTEN-START]
 AN NACHBEDINGUNGEN   (STEHT_AUF ?X ?Y) =  (STEHT_AUF B C)
                                           [KNOTEN-ZIEL]
                      (IST_FREI ?Z)
                      (NICHT (STEHT_AUF ?X ?Z))
                      (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN     KNOTEN-3
ABHAENGIGKEITEN       (IST_FREI ?X)  VON  KNOTEN-START
                      (IST_FREI ?Y)  VON  KNOTEN-START
                      (STEHT_AUF ?X ?Z)  VON  KNOTEN-4
MERKMAL-ERZEUGUNGEN   (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-1:
  ---------
VARIABLEN             (?Y ?X ?Z)
                      ?Y =  B
                      ?X =  A
                      ?Z =  TISCH
OPERATOR              PUTON
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF ?X ?Z) =  (STEHT_AUF A TISCH)
                                           [KNOTEN-START]
                      (IST_FREI ?X) =  (IST_FREI ?Z)
                                        [KNOTEN-3]
                      (IST_FREI ?Y) =  (IST_FREI B)
                                        [KNOTEN-START]
 AN NACHBEDINGUNGEN   (STEHT_AUF ?X ?Y) =  (STEHT_AUF A B)
                                           [KNOTEN-ZIEL]
                      (IST_FREI ?Z)
                      (NICHT (STEHT_AUF ?X ?Z))
                      (NICHT (IST_FREI ?Y))
VORGAENGER-KNOTEN     KNOTEN-2
ABHAENGIGKEITEN       (IST_FREI ?X)  VON  KNOTEN-3
                      (IST_FREI ?Y)  VON  KNOTEN-START
                      (STEHT_AUF ?X ?Z)  VON  KNOTEN-START
MERKMAL-ERZEUGUNGEN   (STEHT_AUF ?X ?Y)  FUER  KNOTEN-ZIEL

  KNOTEN-ZIEL:
  ------------
VARIABLEN
OPERATOR              ZIEL
MERKMAL-BINDUNGEN
 AN VORBEDINGUNGEN    (STEHT_AUF A B) =  (STEHT_AUF ?X ?Y)
                                          [KNOTEN-1]
```

```
                    (STEHT_AUF B C) =  (STEHT_AUF ?X ?Y)
                                        [KNOTEN-2]
                    (STEHT_AUF C D) =  (STEHT_AUF ?X ?Y)
                                        [KNOTEN-3]
 AN NACHBEDINGUNGEN
VORGAENGER-KNOTEN    KNOTEN-1
ABHAENGIGKEITEN      (STEHT_AUF A B)  VON  KNOTEN-1
                     (STEHT_AUF B C)  VON  KNOTEN-2
                     (STEHT_AUF C D)  VON  KNOTEN-3
MERKMAL-ERZEUGUNGEN

-------------------------------------------------------------
```

Die lösende Operator-Sequenz lautet also:

1. Aktion: NEWTOWER (B, D),
2. Aktion: PUTON (C, D, A),
3. Aktion: PUTON (B, C, TISCH),
4. Aktion: PUTON (A, B, TISCH).

LITERATURVERZEICHNIS

[Brooks 1985] : Brooks, R.A., Lozano-Perez,T.
An approach to automatic robot programming, MIT Artificial Intelligence Laboratory, A.I. Memo No. 842 (1985).

[Chapman 1987] : Chapman, D.
Planning for conjunctive goals, Artificial Intelligence 32 (3) (1987) 333-377.

[Fikes 1971] : Fikes, R. E., Nilsson, N. J.
STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (1971) 189 (ff.).

[Hertzberg 1986] : Hertzberg, J.
Planerstellungsmethoden der Künstlichen Intelligenz, Informatik-Spektrum 9 (1986) 149-161.

[Hertzberg 1989] : Hertzberg, J.
Planen: Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz, BI-Wiss.-Verlag (1989).

[Kienzler 1988] : Kienzler, F.
Untersuchung von Planungssystemen, Studienarbeit am Institut für Prozeßrechentechnik und Robotik der Universität Karlsruhe (1988).

[Meyer 1988] : Meyer, O.
Programmieren in COMMON LISP, BI-Wiss.-Verlag (1988).

[Newell 1959] : Newell, A., Shaw, J. C., Simon, H. A.
Report on a general problem solving program, Proc. Int. Conf. on Information Processing (ICIP), Paris (1959).

[Puppe 1988] : Puppe, F.
Einführung in Expertensysteme, Springer Verlag (1988).

[Richter 1989] : Richter, M. M.
Prinzipien der Künstlichen Intelligenz: Wissensrepräsentation, Inferenz und Expertensysteme, Teubner Verlag (1989).

[Sacerdoti 1977] : Sacerdoti, E. D.
A Structure for Plans and Behavior, New York: Elsevier / North-Holland (1977).

[Sussman 1975] : Sussman, G. J.
A Computer Model of Skill Acquisition, New York: Elsevier (1975).

[Schnupp 1986] : Schnupp, P., Leibrandt, U.
Expertensysteme, Springer Verlag (1986).

[Stefik 1981a] : Stefik, M.
Planning with Constraints (MOLGEN Part I), Artificial Intelligence 16 (1981) 111-140.

[Stefik 1981b] : Stefik, M.
Planning and Meta-Planning (MOLGEN Part II), Artificial Intelligence 16 (1981) 141-170.

[Wilkins 1984] : Wilkins, D. E.
Domain-independent planning: representation and plan generation, Artificial Intelligence 22 (1984) 269-301.

[Wilkins 1986] : Wilkins, D. E.
Hierarchical planning: definition and implementation, ECAI (1986) 466-478.

[Winston 1987] : Winston, P. H., Horn, B. K. P.
LISP, Addison Wesley (1987).

Anhang

Programmlisting in COMMON LISP

Implementierung eines nichtlinearen Planungssystems

D I P L O M A R B E I T

von

cand. inform. Friedemann Kienzler

Universität Karlsruhe
Fakultät für Informatik
Institut für Prozeßrechentechnik und Robotik
Prof. Dr.-Ing. U. Rembold
Prof. Dr.-Ing. R. Dillmann

Juni 1990

Referent: Prof. Dr.-Ing. U. Rembold
Korreferent: Prof. Dr.-Ing. R. Dillmann
Betreuer: Dipl.-Inform. A. Köhne

Modulstruktur und Speicherplatzbedarf des implementierten Planers

Planer-Kontrolle.lisp	29.104 Bytes
Planer-Routinen.lisp	137.668 Bytes
Planer-Konfliktbehandlung.lisp	55.080 Bytes
Planer-Duplizierung.lisp	16.751 Bytes
Planer-Ausgabe.lisp	34.169 Bytes
Planer.lisp	272.772 Bytes
Problemstellung.lisp	7.726 Bytes

```
; ********************************************************************
; *   Implementierung eines nichtlinearen Planungsprogramms FKNLP    *
; *           (basierend auf der Beschreibung von TWEAK)             *
; *                                                                  *
; *    Autor:  Friedemann Kienzler                                   *
; *    Stand:  29.05.1990                                            *
; ********************************************************************



; ====================================================================

; ===================       STRUKTUREN      ==========================

; ====================================================================



  (DEFSTRUCT (PLAN)
; --------------------------------------------------------------------
; Was wird gemacht:    Definition der Datenstruktur PLAN
; Globale Variablen:   --
; --------------------------------------------------------------------

      (KNOTENLISTE          NIL)    ; Liste der in PLAN enthalte-
                                    ; nen Plan-Knoten.
      (WERTCONSTRAINTLISTE  NIL))   ; Liste der in PLAN geltenden
                                    ; Wert-Constraints (Codesig-
                                    ; nation- und Noncodesigna-
                                    ; tion-Constraints) direkt
                                    ; zwischen Merkmalen.
                                    ; (Bea.: Diese Plan-Knoten und
                                    ;        Wert-Constraints sind
                                    ;        ebenfalls als Struk-
                                    ;        turen definiert !)

; -PLAN---------------------------------------------------------------



  (DEFSTRUCT (KNOTEN)
; --------------------------------------------------------------------
; Was wird gemacht:    Definition der Datenstruktur KNOTEN
; Globale Variablen:   --
; --------------------------------------------------------------------

      (NAME                 (GENSYM))
                                    ; zur eindeutigen Kennung eines
                                    ; jeden Knotens.
      (VARIABLEN            NIL)    ; Liste der Variablen, die in
                                    ; den Vorbedingungs- oder Nach-
                                    ; bedingungs-Merkmalen der Ope-
                                    ; rator-Instanz vorkommen.
      (OPERATOR             NIL)    ; Name des Operators (z.B. ZIEL
                                    ; oder PICKUP), dessen Wert die
                                    ; eigentliche Operator-Struktur
                                    ; (Vorbedingungen, Nachbedin-
                                    ; gungen, lokale Constraints)
```

```
                                    ; darstellt.
      (CODESIGNATIONS       NIL)    ; Liste saemtlicher Wert-Con-
                                    ; straints (Codesignation-Con-
                                    ; straints), welche direkt an
                                    ; eines der (Vor- oder Nachbe-
                                    ; dingungs-) Merkmale der Ope-
                                    ; rator-Instanz - eventuell
                                    ; implizit mit Restriktion -
                                    ; angelagert sind.
      (NONCODESIGNATIONS    NIL)    ; Liste saemtlicher Wert-Con-
                                    ; straints (Noncodesignation-
                                    ; Constraints), welche direkt
                                    ; an eines der (Vor- oder
                                    ; Nachbedingungs-) Merkmale
                                    ; der Operator-Instanz ange-
                                    ; lagert sind.
                                    ; (Bea.: Wert-Constraints wer-
                                    ;        den ebenfalls durch
                                    ;        Strukturen repraesen-
                                    ;        tiert (s.u.) !)
      (VORGAENGER           NIL)    ; Liste, deren Elemente Ver-
                                    ; weise auf alle die Knoten im
                                    ; Plan darstellen, welche di-
                                    ; rekte Vorgaenger-Knoten von
                                    ; KNOTEN sind (-> Zeit-Con-
                                    ; straints zur Darstellung ei-
                                    ; ner (partiellen) Operator-
                                    ; bzw. Knoten-Ordnung).
      (ist_abhaengig_von    NIL)    ; Liste zweielementiger Unter-
                                    ; listen, deren erstes Element
                                    ; jeweils einen Verweis auf
                                    ; den Knoten im Plan bildet,
                                    ; welcher fuer den KNOTEN sein
                                    ; im zweiten Element der Un-
                                    ; terliste angefuehrte Vorbe-
                                    ; dingungs-Merkmal erzeugt.
      (erzeugt_fuer         NIL))   ; Liste zweielementiger Unter-
                                    ; listen, deren erstes Element
                                    ; jeweils einen Verweis auf
                                    ; den Knoten im Plan bildet,
                                    ; fuer welchen KNOTEN das im
                                    ; zweiten Element der Unter-
                                    ; liste angefuehrte Merkmal
                                    ; (= ein Nachbedingungs-Merk-
                                    ; mal von KNOTEN) erzeugt.

; -KNOTEN-------------------------------------------------------------




  (DEFSTRUCT (OPERATOR)
; ---------------------------------------------------------------------
; Was wird gemacht:    Definition der Datenstruktur OPERATOR
; Globale Variablen:   --
; ---------------------------------------------------------------------

      (VORBEDINGUNGEN       NIL)    ; Liste von Vorbedingungs-
                                    ; Merkmalen.
      (NACHBEDINGUNGEN      NIL)    ; Liste von Nachbedingungs-
                                    ; Merkmalen.
      (LOKALE_CONSTRAINTS   NIL))   ; Liste dreielementiger Unter-
```

```
                                    ; listen (z.B. (<> ?X ?Y) oder
                                    ; (<> ?X A) ), welche "lokale"
                                    ; Wert-Constraints (Noncode-
                                    ; signation-Constraints) be-
                                    ; zueglich OPERATOR darstel-
                                    ; len, mit folgendem Aufbau:
                                    ; 1. Element:  Constraint-Art
                                    ;              <> (redundant),
                                    ; 2. Element:  (erste) Argu-
                                    ;              ment-Variable,
                                    ; 3. Element:  zweite Argu-
                                    ;              ment-Variable
                                    ;              bzw.
                                    ;              Konstante.

; -OPERATOR-----------------------------------------------------------



  (DEFSTRUCT (WERTCONSTRAINT)
; ---------------------------------------------------------------------
; Was wird gemacht:    Definition der Datenstruktur WERTCONSTRAINT
; Globale Variablen:   --
; ---------------------------------------------------------------------

      (NAME                 (GENSYM))
                                    ; zur eindeutigen Kennung eines
                                    ; jeden Wert-Constraints.
      (KNOTEN               NIL)    ; zweielementige Liste, deren
                                    ; Elemente Verweise auf die durch
                                    ; das WERTCONSTRAINT direkt ver-
                                    ; bundenen Plan-Knoten sind.
      (MERKMALPAAR          NIL))   ; zweielementige Liste, deren
                                    ; Elemente die im WERTCONSTRAINT
                                    ; direkt gebundenen beiden Merk-
                                    ; male sind.

; -WERTCONSTRAINT-----------------------------------------------------


; =STRUKTUREN=========================================================




; ====================================================================

; ==================      GLOBALE VARIABLEN      =====================

; ====================================================================



  (DEFUN GLOBALE-VARIABLEN-DEFINIEREN ()
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    NEGATIONSOPERATOR
;                       VERFUEGBARE_OPERATOREN
;                       START
```

```
;                       ZIEL
;
; Eingangsparameter:    --
;
; Nebeneffekte:         Definition der Symbole
;                       NEGATIONSOPERATOR (Wert: 'NICHT),
;                       VERFUEGBARE_OPERATOREN (Wert: Liste
;                             der Namen saemtlicher verfueg-
;                             barer Operatoren (= Operator-
;                             schablonen, nicht Operator-In-
;                             stanzen, welche ja durch die
;                             Planknoten repraesentiert wer-
;                             den), wobei die Namen an die
;                             eigentlichen Operator-Struktu-
;                             ren, die die Operator-Schablo-
;                             nen darstellen, gebunden
;                             sind),
;                       START (Wert: Startoperator-Struktur),
;                       ZIEL (Wert: Zieloperator-Struktur)
;                       als globale Variablen.
;
; ruft auf:             BESTIMME-OPERATOREN
;                       STARTOPERATOR
;                       ZIELOPERATOR
;
; aufgerufen von:       FKNLP
;
; -GLOBALE-VARIABLEN-DEFINIEREN--------------------------------------

     (SETQ
               ; Einfuehrung des Symbols NEGATIONSOPERATOR als globale
               ; Variable:
               NEGATIONSOPERATOR
               'NICHT

               ; Einfuehrung des Symbols VERFUEGBARE_OPERATOREN als
               ; globale Variable:
               VERFUEGBARE_OPERATOREN
               (BESTIMME-OPERATOREN)
                  ; VERFUEGBARE_OPERATOREN ist nun eine Liste,
                  ; welche als Elemente die Namen saemtlicher
                  ; verfuegbarer Operatoren enthaelt, deren
                  ; Werte die eigentlichen Operator-Strukuren
                  ; darstellen.

               ; Einfuehrung des Symbols START als globale Variable:
               START
               (STARTOPERATOR)
                        ; START ist nun die Startoperator-Struktur,
                        ; deren Nachbedingungen die Beschreibung der
                        ; gegebenen Startsituation darstellen.

               ; Einfuehrung des Symbols ZIEL als globale Variable:
               ZIEL
               (ZIELOPERATOR)
                        ; ZIEL ist nun die Zieloperator-Struktur,
                        ; deren Vorbedingungen die Beschreibung der
                        ; erwuenschten Finalsituation darstellen.
     ))

; -GLOBALE-VARIABLEN-DEFINIEREN--------------------------------------
```

```
; =GLOBALE VARIABLEN==================================================




; ====================================================================

; ==================       FUNKTIONEN      ===========================

; ====================================================================


; --------------------------------------------------------------------
; Zunaechst folgen die Prozeduren der obersten Kontrollebene des
; Planers:
;    FKNLP,
;    PLANEN,
;    PLAN-ERWEITERUNG,
;    ERZEUGER-ERFUELLT-MERKMAL,
;    KONFLIKT-LOESUNGEN.
; Diesen schliessen sich dann diesen untergeordnete Prozeduren an,
; welche, in Funktionsbloecke gegliedert, alphabetisch sortiert nach
; Prozedurnamen angefuehrt sind.
; --------------------------------------------------------------------



; ====================================================================
; ==========       OBERSTE KONTROLLEBENE DES PLANERS       ===========
; ====================================================================

; Stand: 29.05.1990


  (DEFUN FKNLP ()
; --------------------------------------------------------------------
; Wert:                 - 'FERTIG, falls fuer das gestellte Planungs-
;                         problem eine Loesung (-> Ausgabe) gefunden
;                         wurde,
;                       - '(UNLOESBARES PLANUNGSPROBLEM!), sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:    --
;
; Nebeneffekte:         Erzeugung unvollstaendiger nichtlinearer Plae-
;                       ne und ggf. Ausgabe eines nichtlinearen Pla-
;                       nes fuer das gestellte Problem.
;
; ruft auf:             GLOBALE-VARIABLEN-DEFINIEREN
;                       ERZEUGE-INITIALPLAN
;                       TEXT-AUSGABE1
;                       PLANEN
;
; aufgerufen von:       Benutzer
;                       (bzw. Top-Level des LISP-Systems)
;
; -FKNLP--------------------------------------------------------------
```

```
      (LOAD "Problemstellung")
                ; Nun sind dem Planer gegebene Start- u. erwuenschte
                ; Finalsituation, sowie die zur Verfuegung stehenden
                ; Operator-Schablonen bekannt.

      (GLOBALE-VARIABLEN-DEFINIEREN)
                ; Nun sind NEGATIONSOPERATOR, VERFUEGBARE_OPERATOREN
                ; und KONSTANTEN als globale Variablen definiert!

      (LET ((INITIALPLAN  (ERZEUGE-INITIALPLAN)))
                ; INITIALPLAN besteht lediglich aus Start- und Ziel-
                ; knoten.

           (TEXT-AUSGABE1)

           (PLANEN (LIST INITIALPLAN))) )
                ; Initiierung der Plan-Generierung fuer das gestell-
                ; te Problem!

; -FKNLP-------------------------------------------------------------




  (DEFUN PLANEN (PLAENE
                 &OPTIONAL (ZAEHLER (- 1)))
; ---------------------------------------------------------------------
; Wert:                 - 'FERTIG, falls fuer das gestellte Planungs-
;                         problem eine Loesung (-> Ausgabe) gefunden
;                         wurde,
;                       - '(UNLOESBARES PLANUNGSPROBLEM!), sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAENE
;                       Plan-Warteschlange.
;                       Liste unvollstaendiger nichtlinearer Plaene,
;                       die gemaess der Breitensuchstrategie zur Er-
;                       mittlung einer Loesung fuer das gestellte Pro-
;                       blem verwaltet bzw. abgearbeitet wird.
;
;                         ZAEHLER (optional!)
;                       Zaehler-Variable fuer die generierten bzw.
;                       aktuell betrachteten nichtlinearen Plaene. Ist
;                       als optionaler Parameter zur Einsparung der
;                       Initialisierung mit -1 beim erstmaligen Aufruf
;                       von FKNLP definiert.
;
; Nebeneffekte:         Erzeugung unvollstaendiger nichtlinearer Plae-
;                       ne und ggf. Ausgabe eines nichtlinearen Pla-
;                       nes fuer das gestellte Problem.
;
; ruft auf:             ERGEBNIS-AUSGABE
;                       FERTIG
;                       PLAN-ERWEITERUNG
;                       WAEHLE-PLAN
;                       TEXT-AUSGABE2
;
; aufgerufen von:       FKNLP
;
; -PLANEN------------------------------------------------------------
```

```
        (COND ((NULL PLAENE) (PRINT '(UNLOESBARES PLANUNGSPROBLEM!)))
              ( T  (LET ((AKTUELLER_PLAN
                                (WAEHLE-PLAN
                                       PLAENE)))
                           (TEXT-AUSGABE2
                                (SETQ ZAEHLER
                                      (+ ZAEHLER 1))
                                (LENGTH (PLAN-KNOTENLISTE
                                                 AKTUELLER_PLAN))
                                (LENGTH PLAENE))
                           (COND ((FERTIG AKTUELLER_PLAN)
                                  (ERGEBNIS-AUSGABE
                                         AKTUELLER_PLAN))
                           ( T  (PLANEN
                                   (NCONC
                                       (DELETE AKTUELLER_PLAN
                                               PLAENE)
                                       (PLAN-ERWEITERUNG
                                               AKTUELLER_PLAN))
                                   ZAEHLER))) ))) )

                    ; Durchsuchen des Suchraums der Plaene via
                    ; Breitensuche.

; -PLANEN-------------------------------------------------------------



  (DEFUN PLAN-ERWEITERUNG (AKTUELLER_PLAN)
; ---------------------------------------------------------------------
; Wert:                 Liste von Plaenen, welchen jeweils eigene in-
;                       terne Speicherbereiche zugeordnet sind und die
;                       sich von AKTUELLER_PLAN darin unterscheiden,
;                       dass die Anzahl eingefuehrter Abhaengigkeiten
;                       (zwischen Nutzer und Erzeuger zur Erfuellung
;                       eines Vorbedingungs-Merkmals) um eins inkre-
;                       mentiert und/oder die Zahl vorhandener poten-
;                       tieller Zerstoerer bezueglich einer bestimmten
;                       Abhaengigkeit innerhalb des jeweiligen Plans
;                       gegenueber AKTUELLER_PLAN um eins dekremen-
;                       tiert worden sind/ist.
;
; Globale Variablen:    --
;
; Eingangsparameter:      AKTUELLER_PLAN
;                       Aktuell betrachteter Plan (erstes Element der
;                       Plan-Warteschlange) mit noch mindestens einem
;                       ungeloesten Knoten.
;
; Nebeneffekte:         Erzeugung (unvollstaendiger) nichtlinearer
;                       Plaene fuer das gestellte Problem.
;
; ruft auf:             WAEHLE-UNGELOESTEN-KNOTEN
;                       SUCHE-UNGELOESTE-KNOTEN
;                       WAEHLE-UNERFUELLTES-MERKMAL
;                       BESTIMME-UNERFUELLTE-MERKMALE
;                       ERMITTLE-ERZEUGER
;                       ERZEUGER-ERFUELLT-MERKMAL
;
; aufgerufen von:       PLANEN
```

```
;
; -PLAN-ERWEITERUNG-----------------------------------------------------

        (LET* ((PLANKNOTENLISTE
                      (PLAN-KNOTENLISTE  AKTUELLER_PLAN))
               (UNGELOESTER_KNOTEN
                      (WAEHLE-UNGELOESTEN-KNOTEN
                             (SUCHE-UNGELOESTE-KNOTEN
                                          PLANKNOTENLISTE)))
               (UNERFUELLTES_MERKMAL
                      (WAEHLE-UNERFUELLTES-MERKMAL
                             (BESTIMME-UNERFUELLTE-MERKMALE
                                          UNGELOESTER_KNOTEN)
                             UNGELOESTER_KNOTEN))
               (K_E_M_TRIPELLISTE
                      (ERMITTLE-ERZEUGER
                             PLANKNOTENLISTE
                             UNGELOESTER_KNOTEN
                             UNERFUELLTES_MERKMAL)))
              (ERZEUGER-ERFUELLT-MERKMAL
                      AKTUELLER_PLAN
                      UNGELOESTER_KNOTEN
                      K_E_M_TRIPELLISTE
                      UNERFUELLTES_MERKMAL)))

; -PLAN-ERWEITERUNG-----------------------------------------------------



  (DEFUN ERZEUGER-ERFUELLT-MERKMAL (PLAN_ALT
                                    NUTZER_ALT
                                    K_E_M_TRIPELLISTE
                                    MERKMAL
                                    &OPTIONAL PLAENE)
; ---------------------------------------------------------------------
; Wert:                 Liste von Plaenen, welchen jeweils eigene in-
;                       terne Speicherbereiche zugeordnet sind und die
;                       sich von PLAN_ALT darin unterscheiden,
;                       dass die Anzahl eingefuehrter Abhaengigkeiten
;                       (zwischen Nutzer und Erzeuger zur Erfuellung
;                       eines Vorbedingungs-Merkmals) um eins inkre-
;                       mentiert und/oder die Zahl vorhandener poten-
;                       tieller Zerstoerer bezueglich einer bestimmten
;                       Abhaengigkeit innerhalb des jeweiligen Plans
;                       gegenueber PLAN_ALT um eins dekremen-
;                       tiert worden sind/ist.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAN_ALT
;                       Aktuell betrachteter Plan (erstes Element der
;                       Plan-Warteschlange) mit noch mindestens einem
;                       ungeloesten Knoten.
;
;                         NUTZER_ALT
;                       Aktuell betrachteter ungeloester Knoten aus
;                       PLAN_ALT, fuer den es einen Erzeuger zu be-
;                       stimmen gilt.
;
;                         K_E_M_TRIPELLISTE
;                       Liste dreielementiger Unterlisten mit folgen-
```

```
;                       dem Aufbau:
;                         1. Element: Kennung
;                                     (= 'ALT, falls der ermittelte
;                                        Erzeuger bereits eine etab-
;                                        lierte Operator-Instanz in
;                                        PLAN_ALT ist;
;                                     (= 'NEU, falls der ermittelte
;                                        Erzeuger noch nicht in
;                                        PLAN_ALT eingebaut ist;
;                                     (= 'NOCH_NICHT_ETABLIERT, falls
;                                        der ermittelte Erzeuger eine
;                                        eine auf Grund eines Zerstoe-
;                                        rers noch nicht etablierte
;                                        Operator-Instanz in PLAN_ALT
;                                        darstellt.),
;                         2. Element: ermittelter Erzeuger,
;                         3. Element: Erzeuger-Nachbedingungs-Merkmal,
;                                     welches mit MERKMAL codesignier-
;                                     bar ist.
;
;                         MERKMAL
;                       Unerfuelltes Vorbedingungs-Merkmal von NUTZER_
;                       ALT, fuer das es einen Erzeuger zu bestimmen
;                       gilt.
;
;                         PLAENE (optional!)
;                       Plaene, die in Form einer Liste als Resultat-
;                       wert von ERZEUGER-ERFUELLT-MERKMAL zurueckge-
;                       geben werden.
;                       PLAENE ist als optionaler Parameter definiert,
;                       um beim erstmaligen Aufruf von PLAN-ERWEITERUNG
;                       die explizite Initialisierung mit NIL einzu-
;                       sparen.
;
; Nebeneffekte:         Erzeugung (unvollstaendiger) nichtlinearer
;                       Plaene fuer das gestellte Problem.
;
; ruft auf:             sich selbst
;                       PLAN-KOPIEREN
;                       KNOTEN-EINBAU
;                       KNOTEN-KOPIEREN
;                       EINTRAEGE-AKTUALISIEREN-E
;                       CONSTRAINT-EINBAU
;                       ABHAENGIGKEITEN-AKTUALISIEREN
;                       KONFLIKT-LOESUNGEN
;
; aufgerufen von:       sich selbst
;                       PLAN-ERWEITERUNG
;
; -ERZEUGER-ERFUELLT-MERKMAL----------------------------------------

        (COND ((NULL K_E_M_TRIPELLISTE) PLAENE)
              ( T  (LET* ((K_E_M_TRIPEL
                                (CAR K_E_M_TRIPELLISTE))
                          (KENNUNG
                                (CAR K_E_M_TRIPEL))
                          (ERZEUGER_ALT
                                (CADR K_E_M_TRIPEL))
                          (EFFEKT
                                (CADDR K_E_M_TRIPEL))
                          (PLAN_NEU
                                (PLAN-KOPIEREN
```

```
                                PLAN_ALT))
                          (NEU
                                (EQUAL KENNUNG
                                       'NEU))
                          (ERZEUGER_NEU
                                (COND (NEU
                                       (KNOTEN-EINBAU
                                            ERZEUGER_ALT
                                            PLAN_NEU))
                                      ( T
                                       (KNOTEN-KOPIEREN
                                            ERZEUGER_ALT
                                            (PLAN-KNOTENLISTE
                                                PLAN_NEU))) ))
                          (KNOTENLISTE_NEU
                                (PLAN-KNOTENLISTE
                                       PLAN_NEU))
                          (NUTZER_NEU
                                (KNOTEN-KOPIEREN
                                       NUTZER_ALT
                                       KNOTENLISTE_NEU)))
                         (COND ((OR
                                   (EQUAL KENNUNG
                                          'ALT)
                                   NEU)
                                (EINTRAEGE-AKTUALISIEREN-E
                                       KNOTENLISTE_NEU
                                       NEU
                                       (CONSTRAINT-EINBAU
                                              ERZEUGER_NEU
                                              EFFEKT
                                              NUTZER_NEU
                                              MERKMAL
                                              PLAN_NEU)
                                       NUTZER_NEU
                                       ERZEUGER_NEU
                                       EFFEKT)))
                         (COND (NEU
                                (ABHAENGIGKEITEN-AKTUALISIEREN
                                       ERZEUGER_NEU
                                       KNOTENLISTE_NEU)))
                         (ERZEUGER-ERFUELLT-MERKMAL
                              PLAN_ALT
                              NUTZER_ALT
                              (CDR K_E_M_TRIPELLISTE)
                              MERKMAL
                              (NCONC
                                  PLAENE
                                  (KONFLIKT-LOESUNGEN
                                         PLAN_NEU
                                         NUTZER_NEU
                                         MERKMAL
                                         ERZEUGER_NEU))) ))) )

; -ERZEUGER-ERFUELLT-MERKMAL----------------------------------------



  (DEFUN KONFLIKT-LOESUNGEN (PLAN
                             NUTZER
                             MERKMAL
```

```
                              ERZEUGER)
; ---------------------------------------------------------------------
; Wert:                   - Falls bezueglich der Abhaengigkeit
;                                ERZEUGER - MERKMAL - NUTZER
;                           mindestens ein potentieller Zerstoerer in
;                           PLAN existiert:
;                             Liste von Plaenen, welchen jeweils eigene
;                             interne Speicherbereiche zugeordnet sind
;                             und die sich von PLAN jeweils darin unter-
;                             scheiden, dass die Anzahl potentieller
;                             Zerstoerer bezueglich der Abhaengigkeit
;                                   ERZEUGER - MERKMAL - NUTZER
;                             um eins verringert ist.
;                         - Sonst:
;                             Liste mit genau einem Plan, welcher sich
;                             von PLAN nur darin unterscheidet, dass
;                             der ERZEUGER bezueglich der Abhaengigkeit
;                                   ERZEUGER - MERKMAL - NUTZER
;                             etabliert, d.h. in der Abhaengigkeiten-
;                             Liste von NUTZER als etablierter (d.h.
;                             durch keinen anderen Knoten gestoerter)
;                             Erzeuger eingetragen worden ist.
;
; Globale Variablen:      --
;
; Eingangsparameter:        PLAN
;                         Plan, dessen Nutzer-Abhaengigkeiten vervoll-
;                         staendigt bzw., bei noch nicht behobenen Kon-
;                         flikten, dessen Zahl bestehender Konflikte um
;                         eins reduziert werden sollen.
;
;                           NUTZER
;                         Aktuell betrachteter ungeloester Knoten in
;                         Plan, fuer den ein Erzeuger zu bestimmen ist
;                         bzw. war.
;
;                           MERKMAL
;                         Aktuell betrachtetes unerfuelltes (Vorbedin-
;                         gungs-)Merkmal von NUTZER, fuer das ein Erzeu-
;                         ger zu bestimmen ist bzw. war.
;
;                           ERZEUGER
;                         Zur Erzeugung der Vorbedingung MERKMAL des un-
;                         geloesten Knotens NUTZER bestimmter (alter
;                         oder neuer) Knoten innerhalb von PLAN.
;
; Nebeneffekte:           - Falls bezueglich der Abhaengigkeit
;                                ERZEUGER - MERKMAL - NUTZER
;                           mindestens ein potentieller Zerstoerer in
;                           PLAN existiert:
;                             Erzeugung einer Liste von Plaenen mit je-
;                             weils eigenem Speicherbereich, die um ei-
;                             nen Konflikt gegenueber PLAN (mittels un-
;                             schiedlicher Konfliktloesungsansaetze)
;                             vermindert sind.
;                         - Sonst:
;                             Vervollstaendigung der Abhaengigkeits-Ein-
;                             traege von NUTZER derart, dass ERZEUGER
;                             als etablierter Produzent von MERKMAL fuer
;                             NUTZER eingetragen wird.
;
; ruft auf:               SUCHE-ZERSTOERER
```

```
;                       ERZEUGER-ETABLIEREN
;                       BESTIMME-STARTKNOTEN
;                       BESTIMME-ZIELKNOTEN
;                       KONFLIKT-ANALYSIEREN
;                       ORDNUNG-VERSCHAERFEN
;                       MERKMALE-NONCODESIGNIERBAR
;                       SEPARIEREN
;                       RETTER-EINSCHIEBEN
;
; aufgerufen von:       ERZEUGER-ERFUELLT-MERKMAL
;
; -KONFLIKT-LOESUNGEN-------------------------------------------------

       (LET* ((PLANKNOTENLISTE
                      (PLAN-KNOTENLISTE
                             PLAN))
              (ZERSTOERER_KMPAAR
                      (SUCHE-ZERSTOERER
                           PLANKNOTENLISTE
                           NUTZER
                           MERKMAL
                           ERZEUGER)))
             (COND ((NULL ZERSTOERER_KMPAAR)
                    (ERZEUGER-ETABLIEREN
                           NUTZER
                           MERKMAL
                           ERZEUGER)
                    (LIST PLAN))
                   ( T  (LET* ((STARTKNOTEN
                                      (BESTIMME-STARTKNOTEN
                                             PLANKNOTENLISTE))
                               (ZIELKNOTEN
                                      (BESTIMME-ZIELKNOTEN
                                             PLANKNOTENLISTE))
                               (ZERSTOERER
                                      (CAR ZERSTOERER_KMPAAR))
                               (ZERSTOERER_EFFEKT
                                      (CADR ZERSTOERER_KMPAAR))
                               (KONFLIKT_ART
                                      (KONFLIKT-ANALYSIEREN
                                             ZERSTOERER
                                             NUTZER
                                             ERZEUGER)))
                              (NCONC
                               (COND ((AND
                                          (OR
                                            (EQUAL KONFLIKT_ART
                                                   'PARALLEL-KONFLIKT)
                                            (EQUAL KONFLIKT_ART
                                                   'GABEL-KONFLIKT_1))
                                          (NOT (EQL ERZEUGER
                                                    STARTKNOTEN)))
                                      (ORDNUNG-VERSCHAERFEN
                                             'VORVERLEGEN
                                             PLAN
                                             ERZEUGER
                                             ZERSTOERER
                                             NUTZER
                                             MERKMAL)))
                                 (COND ((AND
                                          (OR
                                            (EQUAL KONFLIKT_ART
```

```
                                        'PARALLEL-KONFLIKT)
                                  (EQUAL KONFLIKT_ART
                                        'GABEL-KONFLIKT_2))
                                (NOT (EQL NUTZER
                                          ZIELKNOTEN)))
                            (ORDNUNG-VERSCHAERFEN
                                   'NACHVERLEGEN
                                   PLAN
                                   ERZEUGER
                                   ZERSTOERER
                                   NUTZER
                                   MERKMAL)))
                          (COND ((MERKMALE-NONCODESIGNIERBAR
                                   ZERSTOERER
                                   ZERSTOERER_EFFEKT
                                   NUTZER
                                   MERKMAL)
                            (SEPARIEREN
                                   PLAN
                                   ERZEUGER
                                   ZERSTOERER
                                   ZERSTOERER_EFFEKT
                                   NUTZER
                                   MERKMAL)))
                          (RETTER-EINSCHIEBEN
                                 PLAN
                                 ERZEUGER
                                 ZERSTOERER
                                 ZERSTOERER_EFFEKT
                                 NUTZER
                                 MERKMAL))) ))) )

; -KONFLIKT-LOESUNGEN-------------------------------------------------


; =OBERSTE KONTROLLEBENE DES PLANERS==================================
```

```
; ======================================================================
; ===============    ALLGEMEINE PLANUNGSFUNKTIONEN    ================
; ======================================================================

; Stand: 29.05.1990


  (DEFUN ABHAENGIGKEITEN-AKTUALISIEREN (NEUER_KNOTEN
                                        PLANKNOTENLISTE)
; ----------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      NEUER_KNOTEN
;                       Planknoten, der unmittelbar vor Aufruf von
;                       ABHAENGIGKEITEN-AKTUALISIEREN in den aktuell
;                       betrachteten Plan neu eingebaut wurde und der
;                       auf andere bereits bestehende Abhaengigkeiten
;                       als Zerstoerer wirken koennte.
;
;                         PLANKNOTENLISTE
;                       Knotenliste des aktuell betrachteten Plans nach
;                       Einbau eines neuen Knotens, also einschliess-
;                       lich NEUER_KNOTEN.
;
; Nebeneffekte:         Modifizieren der bestehenden Abhaengigkeiten
;                       dahingehend, dass all diejenigen Abhaengigkei-
;                       ten, auf die der neu eingebaute Knoten NEUER_
;                       KNOTEN stoerend wirken koennte, zwar bestehen
;                       bleiben, aber zu nicht etablierten Abhaengig-
;                       keiten transformiert werden, um dann in einem
;                       spaeteren Plan-Schritt durch zusaetzliche Mo-
;                       difikationen erneut etabliert zu werden.
;
;                       (Anm.: Die Abhaengigkeiten werden nur inner-
;                              halb der KNOTEN-ist_abhaengig_von-,
;                              nicht aber innerhalb der -erzeugt_fuer-
;                              Eintraege modifiziert!)
;
; ruft auf:             ABHAENGIGKEITEN-AKTUALISIEREN-HILF
;                       BESTIMME-VORGAENGER
;                       IST-VORGAENGER
;                       BESTIMME-NEGIERTE-EFFEKTE
;
; aufgerufen von:       ERZEUGER-ERFUELLT-MERKMAL
;                       RETTER-NEUTRALISIERT-ZERSTOERER
;
; -ABHAENGIGKEITEN-AKTUALISIEREN-------------------------------------

        (LET ((KNOTEN_VORGAENGER
                    (BESTIMME-VORGAENGER
                              NEUER_KNOTEN)))
             (ABHAENGIGKEITEN-AKTUALISIEREN-HILF
                   NEUER_KNOTEN
                   (BESTIMME-NEGIERTE-EFFEKTE
                          (KNOTEN-OPERATOR
                                NEUER_KNOTEN))
                   (REMOVE-IF
                       #'(LAMBDA (PLANKNOTEN)
                             (IST-VORGAENGER
```

```
                                    PLANKNOTEN
                                    NEUER_KNOTEN
                                    KNOTEN_VORGAENGER))
                          PLANKNOTENLISTE))) )

; -ABHAENGIGKEITEN-AKTUALISIEREN-------------------------------------



  (DEFUN ABHAENGIGKEITEN-AKTUALISIEREN-HILF (NEUER_KNOTEN
                                             KNOTEN_EFFEKTE
                                             PLANKNOTENLISTE)
; ------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      NEUER_KNOTEN
;                       Planknoten, der unmittelbar zuvor in den aktu-
;                       ell betrachteten Plan neu eingebaut wurde und
;                       der auf andere bereits bestehende Abhaengig-
;                       keiten als Zerstoerer wirken koennte.
;
;                         KNOTEN_EFFEKTE
;                       Liste der negierten Nachbedingungs-Merkmale
;                       von NEUER_KNOTEN in nichtnegierter Form.
;
;                         PLANKNOTENLISTE
;                       Knotenliste des aktuell betrachteten Plans
;                       nach Einbau eines neuen Knotens, also ein-
;                       schliesslich NEUER_KNOTEN.
;
; Nebeneffekte:         Modifizieren der bestehenden Abhaengigkeiten
;                       dahingehend, dass all diejenigen Abhaengigkei-
;                       ten, auf die der neu eingebaute Knoten NEUER_
;                       KNOTEN stoerend wirken koennte, zwar bestehen
;                       bleiben, aber zu nicht etablierten Abhaengig-
;                       keiten transformiert werden, um dann in einem
;                       spaeteren Plan-Schritt durch zusaetzliche Mo-
;                       difikationen erneut etabliert zu werden.
;
;                       (Anm.: Die Abhaengigkeiten werden nur inner-
;                              halb der KNOTEN-ist_abhaengig_von-,
;                              nicht aber innerhalb der -erzeugt_fuer-
;                              Eintraege modifiziert!)
;
; ruft auf:             sich selbst
;                       ABHAENGIGKEITEN-PRUEFEN
;                       BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
;
; aufgerufen von:       sich selbst
;                       ABHAENGIGKEITEN-AKTUALISIEREN
;
; -ABHAENGIGKEITEN-AKTUALISIEREN-HILF--------------------------------

        (COND ((NULL PLANKNOTENLISTE))
              ( T  (LET ((BETRACHTETER_KNOTEN
                                (CAR PLANKNOTENLISTE)))
                        (ABHAENGIGKEITEN-PRUEFEN
                              NEUER_KNOTEN
                              KNOTEN_EFFEKTE
```

```
                                BETRACHTETER_KNOTEN
                                (BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
                                          BETRACHTETER_KNOTEN))
                          (ABHAENGIGKEITEN-AKTUALISIEREN-HILF
                                NEUER_KNOTEN
                                KNOTEN_EFFEKTE
                                (CDR PLANKNOTENLISTE))) )))

; -ABHAENGIGKEITEN-AKTUALISIEREN-HILF--------------------------------



  (DEFUN ABHAENGIGKEITEN-PRUEFEN (MOEGLICHER_ZERSTOERER
                                  ZERSTOERER_EFFEKTE
                                  NUTZER
                                  ABHAENGIGKEITEN)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      MOEGLICHER_ZERSTOERER
;                       In den aktuell betrachteten Plan neu eingebau-
;                       ter Knoten, der auf andere, bereits bestehende
;                       Abhaengigkeiten von NUTZER zerstoerend wirken
;                       koennte.
;
;                         ZERSTOERER_EFFEKTE
;                       Liste der negierten Nachbedingungs-Merkmale
;                       von MOEGLICHER_ZERSTOERER in nichtnegierter
;                       Form.
;
;                         NUTZER
;                       Knoten, dessen Abhaengigkeiten daraufhin un-
;                       tersucht werden, ob der neu eingebaute Knoten
;                       MOEGLICHER_ZERSTOERER stoerend auf selbige
;                       wirken koennte.
;
;                         ABHAENGIGKEITEN
;                       Liste der (bisher) etablierten Abhaengigkei-
;                       ten des Knotens NUTZER.
;
; Nebeneffekte:         Modifizieren der bestehenden (bislang etab-
;                       lierten) Abhaengigkeiten des Knotens NUTZER
;                       dahingehend, dass all diejenigen Abhaengigkei-
;                       ten, auf die der neu eingebaute Knoten MOEGLI-
;                       CHER_ZERSTOERER stoerend wirken koennte, zwar
;                       bestehen bleiben, aber zu nicht etablierten
;                       Abhaengigkeiten transformiert werden, um dann
;                       in einem spaeteren Plan-Schritt durch zusaetz-
;                       liche Modifikationen erneut etabliert zu wer-
;                       den.
;
; ruft auf:             sich selbst
;                       IST-VORGAENGER
;                       IST-ZERSTOERER-HILF
;                       ERZEUGER-VORMERKEN
;
; aufgerufen von:       sich selbst
;                       ABHAENGIGKEITEN-AKTUALISIEREN-HILF
;
```

```
; -ABHAENGIGKEITEN-PRUEFEN-----------------------------------------

        (COND ((NULL ABHAENGIGKEITEN))
              ( T  (LET* ((ABHAENGIGKEIT
                                (CAR ABHAENGIGKEITEN))
                          (ERZEUGER
                                (CAR ABHAENGIGKEIT))
                          (MERKMAL
                                (CADR ABHAENGIGKEIT)))
                         (COND ((AND
                                   (NOT (IST-VORGAENGER
                                               MOEGLICHER_ZERSTOERER
                                               ERZEUGER))
                                   (IST-ZERSTOERER-HILF
                                           MOEGLICHER_ZERSTOERER
                                           ZERSTOERER_EFFEKTE
                                           NUTZER
                                           MERKMAL))
                                (SETF (KNOTEN-ist_abhaengig_von
                                                       NUTZER)
                                      (DELETE
                                           ABHAENGIGKEIT
                                           (KNOTEN-ist_abhaengig_von
                                                        NUTZER)
                                           :TEST 'EQUAL))
                                (ERZEUGER-VORMERKEN
                                      NUTZER
                                      MERKMAL
                                      ERZEUGER)))
                         (ABHAENGIGKEITEN-PRUEFEN
                               MOEGLICHER_ZERSTOERER
                               ZERSTOERER_EFFEKTE
                               NUTZER
                               (CDR ABHAENGIGKEITEN))) )))

; -ABHAENGIGKEITEN-PRUEFEN-----------------------------------------



  (DEFUN ALTER (PLANKNOTEN)
; ---------------------------------------------------------------------
; Wert:                 Alter (= Zeitraum seit Einbau in den aktuell
;                       betrachteten Plan) von PLANKNOTEN, dargestellt
;                       durch eine natuerliche Zahl: je kleiner die
;                       Zahl (Minimum: 0), desto laenger liegt der
;                       Einbau von PLANKNOTEN in den aktuell betrach-
;                       teten Plan zurueck.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTEN
;                       Knoten, dessen Alter zu bestimmen ist.
;
;                       (Anm.: Die Altersbestimmung erfolgt anhand des
;                              Knoten-Namens.)
;
; Nebeneffekte:         --
;
; ruft auf:             ZEICHEN-ZU-ZAHL
;
; aufgerufen von:       WAEHLE-UNGELOESTEN-KNOTEN-HILF
```

```
;                       PLAN-AUSGEBEN
;
; -ALTER-------------------------------------------------------------

        (LET ((KNOTENNAME
                    (SYMBOL-NAME
                            (KNOTEN-NAME  PLANKNOTEN))) )
             (DO* ((I       0       (+ I 1))
                   (SUMME   0       (+ (ZEICHEN-ZU-ZAHL
                                         (AREF KNOTENNAME I))
                                       (* SUMME 10))) )
                  ((= I
                      (- (LENGTH KNOTENNAME) 1))
                   SUMME))) )

; -ALTER-------------------------------------------------------------




  (DEFUN ANZAHL-ARGUMENTE (MERKMAL)
; --------------------------------------------------------------------
; Wert:                 Zahl der Argumente (= Variablen und Konstan-
;                       ten), die in dem nichtnegierten MERKMAL vor-
;                       kommen.
;                       (Bea.: NIL wird nicht als Argument, also weder
;                              als Variable, noch als Konstante be-
;                              trachtet!)
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMAL
;                       Nichtnegiertes Merkmal, dessen Argumente zu
;                       zaehlen sind.
;
; Nebeneffekte:         --
;
; ruft auf:             ANZAHL-ARGUMENTE-HILF
;
; aufgerufen von:       BILDE-NONCOD-KAPAARE
;                       WAEHLE-UNERFUELLTES-MERKMAL
;
; -ANZAHL-ARGUMENTE--------------------------------------------------

     (ANZAHL-ARGUMENTE-HILF
                 (CDR MERKMAL)))

; -ANZAHL-ARGUMENTE--------------------------------------------------




  (DEFUN ANZAHL-ARGUMENTE-HILF (MERKMALARGUMENTE)
; --------------------------------------------------------------------
; Wert:                 Zahl der Argumente (= Variablen und Konstan-
;                       ten), die in der Argument-Liste MERKMALARGU-
;                       MENTE vorkommen.
;                       (Bea.: NIL wird nicht als Argument, also weder
;                              als Variable, noch als Konstante be-
;                              trachtet!)
;
; Globale Variablen:    --
```

```
;
; Eingangsparameter:      MERKMALARGUMENTE
;                       Nichtnegiertes Merkmal ohne erstes Element,
;                       also lediglich die Merkmalargumente, die zu
;                       zaehlen sind.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       ANZAHL-ARGUMENTE
;
; -ANZAHL-ARGUMENTE-HILF----------------------------------------------

       (COND ((NULL MERKMALARGUMENTE) 0)
             ( T  (+ (IF (NOT (NULL (CAR MERKMALARGUMENTE)))
                         1
                         0)
                     (ANZAHL-ARGUMENTE-HILF
                             (CDR MERKMALARGUMENTE))) )))

; -ANZAHL-ARGUMENTE-HILF----------------------------------------------



  (DEFUN ANZAHL-CODESIGNIERTER-ARGUMENTE (KNOTEN1
                                          MERKMAL1
                                          KNOTEN2
                                          MERKMAL2)
; ------------------------------------------------------------------
; Wert:                 Anzahl der mit entsprechenden Argumenten von
;                       MERKMAL2 codesignierten Argumente von MERK-
;                       MAL1.
;
;                       (Anm.: Das Argument NIL wird hierbei nicht als
;                              codesigniertes Argument mitgezaehlt,
;                              auch wenn beide entsprechenden Argumen-
;                              te NIL sind.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMAL1 (-> KNOTEN1), MERKMAL2 (-> KNOTEN2)
;                       Die Zahl codesignierter Argumente (<> NIL)
;                       dieses Merkmalpaars soll bestimmt werden.
;
; Nebeneffekte:         --
;
; ruft auf:             ANZAHL-CODESIGNIERTER-ARGUMENTE-HILF
;
; aufgerufen von:       BILDE-NONCOD-KAPAARE
;
; -ANZAHL-CODESIGNIERTER-ARGUMENTE-----------------------------------

       (ANZAHL-CODESIGNIERTER-ARGUMENTE-HILF
                  KNOTEN1
                  (CDR MERKMAL1)
                  KNOTEN2
                  (CDR MERKMAL2)))

; -ANZAHL-CODESIGNIERTER-ARGUMENTE-----------------------------------
```

```
  (DEFUN ANZAHL-CODESIGNIERTER-ARGUMENTE-HILF (KNOTEN1
                                               MERKMALARGUMENTE1
                                               KNOTEN2
                                               MERKMALARGUMENTE2)
; ---------------------------------------------------------------------
; Wert:                 Anzahl der mit entsprechenden Argumenten von
;                       MERKMALARGUMENTE2 codesignierten Argumente von
;                       MERKMALARGUMENTE1.
;
;                       (Anm.: Das Argument NIL wird hierbei nicht als
;                              codesigniertes Argument mitgezaehlt,
;                              auch wenn beide entsprechenden Argumen-
;                              te NIL sind.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMALARGUMENTE1 (-> KNOTEN1),
;                         MERKMALARGUMENTE2 (-> KNOTEN2)
;                       Die Zahl codesignierter Argumente (<> NIL)
;                       dieses Paares von Merkmalargument-Listen soll
;                       bestimmt werden.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       ARGUMENTE-CODESIGNIERT
;
; aufgerufen von:       sich selbst
;                       ANZAHL-CODESIGNIERTER-ARGUMENTE
;
; -ANZAHL-CODESIGNIERTER-ARGUMENTE-HILF-------------------------------

       (COND ((NULL MERKMALARGUMENTE1) 0)
             ( T  (+ (IF (ARGUMENTE-CODESIGNIERT
                              KNOTEN1
                              (CAR MERKMALARGUMENTE1)
                              KNOTEN2
                              (CAR MERKMALARGUMENTE2))
                         1
                         0)
                     (ANZAHL-CODESIGNIERTER-ARGUMENTE-HILF
                            KNOTEN1
                            (CDR MERKMALARGUMENTE1)
                            KNOTEN2
                            (CDR MERKMALARGUMENTE2))) )))

; -ANZAHL-CODESIGNIERTER-ARGUMENTE-HILF-------------------------------



  (DEFUN ARGUMENTE-CODESIGNIERT (KNOTEN1
                                 ARGUMENT1
                                 KNOTEN2
                                 ARGUMENT2)
; ---------------------------------------------------------------------
; Wert:                 - T, falls beide Argumente (ARGUMENT1 und AR-
;                         GUMENT2) Konstanten oder beide NIL sind.
```

```
;                       - Liste zweielementiger Unterlisten (Knoten-
;                         Argument-Paare) (also ein Wert <> NIL), wo-
;                         bei das erste Element das Knoten-Argument-
;                         Paar (KNOTEN2 ARGUMENT2) ist, falls die bei-
;                         den Argumente ARGUMENT1 und ARGUMENT2 code-
;                         signieren, d.h. falls das Knoten-Argument-
;                         Paar (KNOTEN2 ARGUMENT2) Element der transi-
;                         tiven Huelle des Knoten-Argument-Paares
;                         (KNOTEN1 ARGUMENT1) bezueglich der Codesig-
;                         nations-Relation ist.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ARGUMENT1 (-> KNOTEN1),
;                         ARGUMENT2 (-> KNOTEN2)
;                       Argumentpaar, das auf norwendigerweise beste-
;                       hende, aber nicht unbedingt explizite Codesig-
;                       niertheit zu untersuchen ist.
;
; Nebeneffekte:         --
;
; ruft auf:             IST-VARIABLE
;                       BESTIMME-ACOD-HUELLE
;
; aufgerufen von:       ANZAHL-CODESIGNIERTER-ARGUMENTE-HILF
;                       ARGUMENTE-NONCODESIGNIERT
;                       MERKMALE-NONCODESIGNIERBAR
;
; -ARGUMENTE-CODESIGNIERT-------------------------------------------

       (COND ((AND
                  (EQUAL ARGUMENT1
                         ARGUMENT2)
                  (NOT (IST-VARIABLE
                               ARGUMENT1)))
              T)
             ((AND
                  (NOT (EQUAL ARGUMENT1
                              ARGUMENT2))
                  (OR
                     (NULL ARGUMENT1)
                     (NULL ARGUMENT2)
                     (AND
                        (NOT (IST-VARIABLE
                                    ARGUMENT1))
                        (NOT (IST-VARIABLE
                                    ARGUMENT2))) ))
              NIL)
             ( T  (MEMBER (LIST KNOTEN2
                                ARGUMENT2)
                          (BESTIMME-ACOD-HUELLE
                                (LIST
                                    (LIST KNOTEN1
                                          ARGUMENT1)))
                          :TEST 'EQUAL))) )

; -ARGUMENTE-CODESIGNIERT-------------------------------------------
```

```
  (DEFUN ARGUMENTE-NONCODESIGNIERT (KNOTEN1
                                    ARGUMENT1
                                    KNOTEN2
                                    ARGUMENT2)
; ---------------------------------------------------------------------
; Wert:                 - T, falls ARGUMENT1 und ARGUMENT2 mit Kon-
;                         stanten codesignieren, die nicht identisch
;                         sind.
;
;                         (Anm.: Fuer zwei verschiedene Konstanten
;                                gilt immer, dass sie notwendigerweise
;                                noncodesignieren.)
;
;                       - Liste zweielementiger Unterlisten (Knoten-
;                         Argument-Paare) (also ein Wert <> NIL), wo-
;                         bei das erste Element das Knoten-Argument-
;                         Paar (KNOTEN2 ARGUMENT2) ist, falls die bei-
;                         den Argumnte ARGUMENT1 und ARGUMENT2 notwen-
;                         digerweise noncodesignieren.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ARGUMENT1 (-> KNOTEN1),
;                         ARGUMENT2 (-> KNOTEN2)
;                       Argument-Paar, das auf notwendigerweise beste-
;                       hende, aber nicht unbedingt explizite Noncode-
;                       signiertheit zu untersuchen ist.
;
;                       (Anm.: Wenn ARGUMENTE-NONCODESIGNIERT aufgeru-
;                              fen wird, ist mindestens eines der bei-
;                              den Argumente ARGUMENT1, ARGUMENT2 eine
;                              Variable.)
;
; Nebeneffekte:         --
;
; ruft auf:             BESTIMME-ACOD-HUELLE
;                       ARGUMENTE-CODESIGNIERT
;                       KAPAARE-ORDNEN
;                       BESTIMME-DIREKTE-ABINDUNGEN
;
; aufgerufen von:       MERKMALE-CODESIGNIERBAR-HILF
;
; -ARGUMENTE-NONCODESIGNIERT-------------------------------------------

        (AND
           (NOT (ARGUMENTE-CODESIGNIERT
                       KNOTEN1
                       ARGUMENT1
                       KNOTEN2
                       ARGUMENT2))
           (LET* ((KAPAAR1
                       (LIST KNOTEN1
                             ARGUMENT1))
                  (KAPAAR2
                       (LIST KNOTEN2
                             ARGUMENT2))
                  (ACODESIGNATIONS1
                       (BESTIMME-ACOD-HUELLE
                           (LIST KAPAAR1)))
                  (ACODESIGNATIONS2
                       (BESTIMME-ACOD-HUELLE
```

```
                        (LIST KAPAAR2)))
                    (CODESIGNIERTE_KONSTANTE1
                        (CADAAR (KAPAARE-ORDNEN
                                ACODESIGNATIONS1)))
                    (CODESIGNIERTE_KONSTANTE2
                        (CADAAR (KAPAARE-ORDNEN
                                ACODESIGNATIONS2)))
                    (ANONCODESIGNATIONS1
                        (BESTIMME-ACOD-HUELLE
                            (BESTIMME-DIREKTE-ABINDUNGEN
                                ACODESIGNATIONS1
                                'KNOTEN-NONCODESIGNATIONS)))
                    (ANONCODESIGNATIONS2
                        (BESTIMME-ACOD-HUELLE
                            (BESTIMME-DIREKTE-ABINDUNGEN
                                ACODESIGNATIONS2
                                'KNOTEN-NONCODESIGNATIONS)))
                    (NONCODESIGNIERTE_KONSTANTEN1
                        (MAPCAR
                            #'(LAMBDA (KKPAAR)
                                  (CADR KKPAAR))
                            (CAR (KAPAARE-ORDNEN
                                ANONCODESIGNATIONS1))) )
                    (NONCODESIGNIERTE_KONSTANTEN2
                        (MAPCAR
                            #'(LAMBDA (KKPAAR)
                                  (CADR KKPAAR))
                            (CAR (KAPAARE-ORDNEN
                                ANONCODESIGNATIONS2))) ))
                   (OR
                      (AND
                         (NOT (NULL CODESIGNIERTE_KONSTANTE1))
                         (NOT (NULL CODESIGNIERTE_KONSTANTE2))
                         (NOT (EQUAL CODESIGNIERTE_KONSTANTE1
                                CODESIGNIERTE_KONSTANTE2)))
                      (AND
                         (NOT (NULL CODESIGNIERTE_KONSTANTE1))
                         (NOT (NULL NONCODESIGNIERTE_KONSTANTEN2))
                         (MEMBER CODESIGNIERTE_KONSTANTE1
                                 NONCODESIGNIERTE_KONSTANTEN2))
                      (AND
                         (NOT (NULL CODESIGNIERTE_KONSTANTE2))
                         (NOT (NULL NONCODESIGNIERTE_KONSTANTEN1))
                         (MEMBER CODESIGNIERTE_KONSTANTE2
                                 NONCODESIGNIERTE_KONSTANTEN1))
                      (MEMBER KAPAAR2
                              ANONCODESIGNATIONS1
                              :TEST 'EQUAL))) ))

; -ARGUMENTE-NONCODESIGNIERT---------------------------------------



  (DEFUN BESTIMME-ACOD-HUELLE (KAPAAR_LISTE
                               &OPTIONAL MARKIERTE_KAPAARE)
; ---------------------------------------------------------------------
; Wert:                Liste zweielementiger Unterlisten (Knoten-Ar-
;                      gument-Paare) mit folgendem Aufbau:
;                        1. Element: Knoten,
;                        2. Element: Argument (Var. oder Konst.).
;                      In dieser Liste sind alle Argumente (mit zuge-
```

```
;                       hoerigem Knoten zur eindeutigen Kennzeichnung)
;                       vertreten, die mit mindestens einem Argument
;                       aus der KAPAAR_LISTE codesignieren.
;                       Die Resultatliste bildet daher die transitive
;                       Huelle saemtlicher Knoten-Argument-Paare aus
;                       KAPAAR_LISTE bezueglich der Codesignations-Re-
;                       lation.
;
; Globale Variablen:    --
;
; Eingangsparameter:      KAPAAR_LISTE
;                       Knoten-Argument-Paarliste, zu der die transi-
;                       tive Huelle bezueglich der Codesignations-Re-
;                       lation zu bestimmen ist.
;
;                         MARKIERTE_KAPAARE (optional!)
;                       Liste der bereits betrachteten Knoten-Argu-
;                       ment-Paare. Ist als optionaler Parameter defi-
;                       niert, um eine explizite Initialisierung mit
;                       NIL beim erstmaligen Aufruf von aussen mit
;                       derselben Wirkung einzusparen.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       BESTIMME-DIREKTE-ABINDUNGEN
;
; aufgerufen von:       sich selbst
;                       ARGUMENTE-CODESIGNIERT
;                       ARGUMENTE-NONCODESIGNIERT
;                       BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF
;                       VARIABLENBINDUNGEN-AUSGEBEN
;
; -BESTIMME-ACOD-HUELLE-----------------------------------------------

       (LET ((NICHTMARKIERTE_KAPAARE
                   (REMOVE-IF
                      #'(LAMBDA (KAPAAR)
                            (MEMBER KAPAAR
                                    MARKIERTE_KAPAARE
                                    :TEST 'EQUAL))
                      KAPAAR_LISTE)))
            (COND ((NULL NICHTMARKIERTE_KAPAARE) MARKIERTE_KAPAARE)
                  ( T  (BESTIMME-ACOD-HUELLE
                           (BESTIMME-DIREKTE-ABINDUNGEN
                                  NICHTMARKIERTE_KAPAARE
                                  'KNOTEN-CODESIGNATIONS)
                           (NCONC
                                  NICHTMARKIERTE_KAPAARE
                                  MARKIERTE_KAPAARE))) )))

; -BESTIMME-ACOD-HUELLE-----------------------------------------------



  (DEFUN BESTIMME-DIREKTE-ABINDUNGEN (KAPAAR_LISTE
                                      BINDUNGSART)
; ---------------------------------------------------------------------
; Wert:                 - Liste zweielementiger Unterlisten (Knoten-
;                         Argument-Paare) mit folgendem Aufbau:
;                           1. Element: Knoten,
```

```
;                          2. Element: Argument (Var. oder Konst.).
;                        In dieser Liste sind alle Argumente (mit zu-
;                        gehoerigem Knoten zur eindeutigen Kennzeich-
;                        nung) vertreten, die mit mindestens einem
;                        Argument aus der KAPAAR_LISTE explizit
;                        (non-)codesignieren.
;                      - NIL, falls an keinem Knoten der KAPAAR_LISTE
;                        ein Codesignation- oder Noncodesignation-
;                        Wertconstraint (je nach BINDUNGSART) angela-
;                        gert ist.
;
; Globale Variablen:   --
;
; Eingangsparameter:     KAPAAR_LISTE
;                      Knoten-Argument-Paar-Liste, zu der die Liste
;                      der expliziten (oder direkten) (Non-)Codesig-
;                      nation-Partner-Paare zu bestimmen ist.
;
;                        BINDUNGSART
;                      Indikator dafuer, ob die expliziten Codesigna-
;                      tion- oder Noncodesignation-Bindungen betrach-
;                      tet werden sollen. Je nachdem ist der Wert von
;                      BINDUNGSART 'KNOTEN-CODESIGNATIONS oder
;                      'KNOTEN-NONCODESIGNATIONS.
;
; Nebeneffekte:        --
;
; ruft auf:            sich selbst
;                      MEHRFACHE-RAUS
;                      BILDE-KAPAARE
;
; aufgerufen von:      sich selbst
;                      ARGUMENTE-NONCODESIGNIERT
;                      BESTIMME-ACOD-HUELLE
;                      VARIABLENBINDUNGEN-AUSGEBEN
;
; -BESTIMME-DIREKTE-ABINDUNGEN----------------------------------------

       (COND ((NULL KAPAAR_LISTE) NIL)
             ( T  (LET* ((K
                              (CAAR KAPAAR_LISTE))
                         (A
                              (CADAR KAPAAR_LISTE))
                         (CONSTRAINTS
                              (FUNCALL BINDUNGSART
                                       K)))
                        (MEHRFACHE-RAUS
                              (APPEND
                                   (BILDE-KAPAARE
                                          CONSTRAINTS
                                          BINDUNGSART
                                          K
                                          A)
                                   (BESTIMME-DIREKTE-ABINDUNGEN
                                          (CDR KAPAAR_LISTE)
                                          BINDUNGSART))) ))) )

; -BESTIMME-DIREKTE-ABINDUNGEN----------------------------------------
```

```
  (DEFUN BESTIMME-DIREKTE-VORGAENGER (AKTUELLE_GENERATION)
; ---------------------------------------------------------------------
; Wert:                 Liste saemtlicher direkter Vorgaenger-Knoten
;                       jedes in AKTUELLE_GENERATION angefuehrten Kno-
;                       tens, wobei solche Knoten, die direkte Vor-
;                       gaenger mehrerer Knoten aus AKTUELLE_GENERATI-
;                       ON sind, jeweils nur einmal in der resultie-
;                       renden Liste erscheinen.
;
; Globale Variablen:    --
;
; Eingangsparameter:      AKTUELLE_GENERATION
;                       Ein Knoten (Atom) bzw. eine Menge von Knoten
;                       (Liste), dessen bzw. deren direkte Vorgaenger
;                       bestimmt werden sollen.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       MEHRFACHE-RAUS
;
; aufgerufen von:       sich selbst
;                       BESTIMME-VORGAENGER
;                       IST-VORGAENGER
;                       IST-VORGAENGER-HILF
;                       SUCHE-ALTE-ERZEUGER
;                       SUCHE-ALTE-ERZEUGER-HILF
;                       SUCHE-UNGELOESTE-KNOTEN
;                       SUCHE-UNGELOESTE-KNOTEN-HILF
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF
;                       SUCHE-ALTE-RETTER
;                       SUCHE-ALTE-RETTER-HILF
;                       SUCHE-ZERSTOERER
;                       SUCHE-ZERSTOERER-HILF
;                       PLAN-AUSGEBEN
;                       PLAN-AUSGEBEN-HILF
;
; -BESTIMME-DIREKTE-VORGAENGER----------------------------------------

        (COND ((NULL AKTUELLE_GENERATION) NIL)
              ((ATOM AKTUELLE_GENERATION)
               (KNOTEN-VORGAENGER AKTUELLE_GENERATION))
              ( T  (LET ((BETRACHTETER_KNOTEN
                                (CAR AKTUELLE_GENERATION)))
                        (MEHRFACHE-RAUS
                            (APPEND
                                (KNOTEN-VORGAENGER
                                    BETRACHTETER_KNOTEN)
                                (BESTIMME-DIREKTE-VORGAENGER
                                    (CDR AKTUELLE_GENERATION))) ))) ))

; -BESTIMME-DIREKTE-VORGAENGER----------------------------------------




  (DEFUN BESTIMME-ETABLIERTE-ABHAENGIGKEITEN (PLANKNOTEN)
; ---------------------------------------------------------------------
; Wert:                 Liste saemtlicher etablierter Abhaengigkeiten
;                       (Eintrag: KNOTEN-ist_abhaengig_von) von
;                       NUTZER.
```

```
;
;                       (Anm.: - Eine Abhaengigkeit ist dann etab-
;                                liert, wenn sie durch keinen anderen
;                                Planknoten moeglicherweise gestoert
;                                werden koennte.
;                              - Eine noch nicht oder nicht mehr etab-
;                                lierte Abhaengigkeit ist als solche
;                                an folgendem Aufbau zu erkennen:
;                       -->       1. Element: 'NOCH_NICHT_ETABLIERT,
;                                 2. Element: Erzeuger-Knoten,
;                                 3. Element: erzeugte Vorbedingung.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTEN
;                       Planknoten, dessen etablierte Abhaengigkeiten
;                       zu bestimmen sind.
;
; Nebeneffekte:         --
;
; ruft auf:             --
;
; aufgerufen von:       ABHAENGIGKEITEN-AKTUALISIEREN-HILF
;                       BESTIMME-UNERFUELLTE-MERKMALE
;                       GELOEST
;                       KNOTEN-AUSGEBEN
;
; -BESTIMME-ETABLIERTE-ABHAENGIGKEITEN-------------------------------

       (REMOVE-IF
           #'(LAMBDA (ABHAENGIGKEIT)
                  (= (LENGTH ABHAENGIGKEIT)
                     3))
           (KNOTEN-ist_abhaengig_von
                           PLANKNOTEN)))

; -BESTIMME-ETABLIERTE-ABHAENGIGKEITEN-------------------------------



  (DEFUN BESTIMME-NEGIERTE-EFFEKTE (OPERATORNAME)
; ------------------------------------------------------------------
; Wert:                 Liste saemtlicher negierter Nachbedingungs-
;                       Merkmale der Operator-Struktur, die an OPERA-
;                       TORNAME als Wert begunden ist, in nichtnegier-
;                       ter Form.
;
; Globale Variablen:    --
;
; Eingangsparameter:      OPERATORNAME
;                       Bezeichnung des Operators, dessen negierte
;                       Nachbedingungs-Merkmale in nichtnegierter Form
;                       bestimmt werden sollen.
;
; Nebeneffekte:         --
;
; ruft auf:             IST-NEGIERT
;
; aufgerufen von:       ABHAENGIGKEITEN-AKTUALISIEREN
;                       IST-ZERSTOERER
;
```

```
; -BESTIMME-NEGIERTE-EFFEKTE----------------------------------------

        (MAPCAR
            #'(LAMBDA (EFFEKT)
                  (CADR EFFEKT))
            (REMOVE-IF-NOT
                'IST-NEGIERT
                (OPERATOR-NACHBEDINGUNGEN
                      (EVAL OPERATORNAME))) ))

; -BESTIMME-NEGIERTE-EFFEKTE----------------------------------------



  (DEFUN BESTIMME-NICHTNEGIERTE-EFFEKTE (OPERATORNAME)
; ------------------------------------------------------------------
; Wert:                 Liste saemtlicher nichtnegierter Nachbedin-
;                       gungs-Merkmale der Operator-Struktur, die an
;                       OPERATORNAME als Wert begunden ist.
;
; Globale Variablen:    --
;
; Eingangsparameter:      OPERATORNAME
;                       Bezeichnung des Operators, dessen nichtnegier-
;                       te Nachbedingungs-Merkmale bestimmt werden
;                       sollen.
;
; Nebeneffekte:         --
;
; ruft auf:             IST-NEGIERT
;
; aufgerufen von:       HOLE-NEUE-ERZEUGER
;                       IST-ERZEUGER
;
; -BESTIMME-NICHTNEGIERTE-EFFEKTE-----------------------------------

        (REMOVE-IF
            'IST-NEGIERT
            (OPERATOR-NACHBEDINGUNGEN
                  (EVAL OPERATORNAME))) )

; -BESTIMME-NICHTNEGIERTE-EFFEKTE-----------------------------------



  (DEFUN BESTIMME-OPERATOREN ()
; ------------------------------------------------------------------
; Wert:                 Liste der Namen saemtlicher verfuegbarer Ope-
;                       ratoren, welche gemaess der Anzahl der Vorbe-
;                       dingungs-Merkmale der entsprechenden Operator-
;                       Struktur aufsteigend sortiert sind: der Name
;                       des Operators mit den wenigsten Vorbedingungen
;                       steht an erster Stelle. Die Werte der Opera-
;                       tor-Namen stellen die eigentliche Operator-
;                       Struktur (mit Vor- und Nachbedingungen und
;                       lokalen Constraints) dar.
;
; Globale Variablen:    --
;
; Eingangsparameter:    --
```

```
;
; Nebeneffekte:         Ausgehend von der Operator-Spezifikation (sie-
;                       he Funktion OPERATOREN) werden an die einzel-
;                       nen Operator-Namen als Werte die eigentlichen
;                       Operator-Strukturen (= Operator-Schablonen,
;                       die instantiiert werden koennen) global gebun-
;                       den.
;
; ruft auf:             OPERATOREN
;
; aufgerufen von:       GLOBALE-VARIABLEN-DEFINIEREN
;
; -BESTIMME-OPERATOREN-----------------------------------------------

        (DO* ((OPSCHABLONEN      (OPERATOREN)
                                 (CDDR OPSCHABLONEN))
              (OPERATORNAME      (CAR OPSCHABLONEN)
                                 (CAR OPSCHABLONEN))
              (OPERATORDEF       (CADR OPSCHABLONEN)
                                 (CADR OPSCHABLONEN))
              (OPLISTE           NIL))
             ((NULL OPSCHABLONEN)
              (SORT OPLISTE
                    #'(LAMBDA (OP1 OP2)
                          (< (LENGTH (OPERATOR-VORBEDINGUNGEN
                                          (EVAL OP1)))
                             (LENGTH (OPERATOR-VORBEDINGUNGEN
                                          (EVAL OP2))) ))) )
             (SETQ OPLISTE
                   (CONS OPERATORNAME OPLISTE))
             (SET OPERATORNAME
                  (EVAL OPERATORDEF))) )

; -BESTIMME-OPERATOREN-----------------------------------------------



  (DEFUN BESTIMME-STARTKNOTEN (PLANKNOTENLISTE)
; ------------------------------------------------------------------
; Wert:                 Startknoten des aktuell betrachteten Plans.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTENLISTE
;                       Knotenliste des aktuell betrachteten Plans.
;
; Nebeneffekte:         --
;
; ruft auf:             --
;
; aufgerufen von:       KONFLIKT-LOESUNGEN
;                       EINTRAEGE-AKTUALISIEREN-E
;                       SUCHE-ALTE-RETTER
;
; -BESTIMME-STARTKNOTEN----------------------------------------------

        (CADR (REVERSE PLANKNOTENLISTE)))

; -BESTIMME-STARTKNOTEN----------------------------------------------
```

```
  (DEFUN BESTIMME-UNERFUELLTE-MERKMALE (UNGELOESTER_KNOTEN)
; ---------------------------------------------------------------------
; Wert:                 Liste saemtlicher unerfuellter Vorbedingungs-
;                       Merkmale der Operator-Instanz UNGELOESTER_
;                       KNOTEN.
;
;                       (Anm.: Eine Vorbedingung ist unerfuellt gdw.
;                              bezueglich ihr keine Abhaengigkeit
;                              etabliert ist.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      UNGELOESTER_KNOTEN
;                       Aktuell betrachteter ungeloester Planknoten im
;                       aktuell betrachteten Plan, dessen unerfuellte
;                       Vorbedingungs-Merkmale zu bestimmen sind.
;
; Nebeneffekte:         --
;
; ruft auf:             BESTIMME-UNERFUELLTE-MERKMALE-HILF
;                       BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
;
; aufgerufen von:       PLAN-ERWEITERUNG
;
; -BESTIMME-UNERFUELLTE-MERKMALE--------------------------------------

      (BESTIMME-UNERFUELLTE-MERKMALE-HILF
                    (OPERATOR-VORBEDINGUNGEN
                           (EVAL (KNOTEN-OPERATOR
                                       UNGELOESTER_KNOTEN)))
                    (BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
                           UNGELOESTER_KNOTEN)))

; -BESTIMME-UNERFUELLTE-MERKMALE--------------------------------------



  (DEFUN BESTIMME-UNERFUELLTE-MERKMALE-HILF (VORBEDINGUNGEN
                                             ABHAENGIGKEITEN)
; ---------------------------------------------------------------------
; Wert:                 Liste saemtlicher unerfuellter Vorbedingungen
;                       aus der Merkmalliste VORBEDINGUNGEN.
;
; Globale Variablen:    --
;
; Eingangsparameter:      VORBEDINGUNGEN
;                       Liste saemtlicher Vorbedingungen des aktuell
;                       betrachteten ungeloesten Planknotens.
;
;                       ABHAENGIGKEITEN
;                       Liste saemtlicher etablierter Abhaengigkeiten
;                       des aktuell betrachteten ungeloesten Plankno-
;                       tens.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
```

```
;                       BESTIMME-UNERFUELLTE-MERKMALE
;
; -BESTIMME-UNERFUELLTE-MERKMALE-HILF-------------------------------

        (COND ((NULL ABHAENGIGKEITEN) VORBEDINGUNGEN)
              ( T  (LET ((ABHAENGIGKEITSMERKMAL
                              (CADAR ABHAENGIGKEITEN)))
                         (BESTIMME-UNERFUELLTE-MERKMALE-HILF
                                (REMOVE-IF
                                   #'(LAMBDA (VORBEDINGUNG)
                                         (EQL VORBEDINGUNG
                                              ABHAENGIGKEITSMERKMAL))
                                   VORBEDINGUNGEN)
                                (CDR ABHAENGIGKEITEN))) )))

; -BESTIMME-UNERFUELLTE-MERKMALE-HILF-------------------------------



  (DEFUN BESTIMME-VARIABLEN (SITUATION
                             &OPTIONAL VARIABLENLISTE)
; ------------------------------------------------------------------
; Wert:                 Liste saemtlicher in SITUATION vorkommender
;                       Variablen, wobei in SITUATION mehrfach vorhan-
;                       dene Variablen in der resultierenden Liste nur
;                       jeweils einmal angefuehrt werden.
;
; Globale Variablen:    NEGATIONSOPERATOR
;
; Eingangsparameter:      SITUATION
;                       Liste von Merkmalen, welche eine bestimmte Si-
;                       tuation spezifizieren.
;
;                         VARIABLENLISTE (optional!)
;                       Liste der bisher in SITUATION entdeckten Va-
;                       riablen. Ist als optionaler Parameter defi-
;                       niert, um eine Initialisierung mit NIL beim
;                       erstmaligen Aufruf von aussen mit derselben
;                       Wirkung einsparen zu koennen.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       MEHRFACHE-RAUS
;                       IST-VARIABLE
;
; aufgerufen von:       sich selbst
;                       ERZEUGE-INITIALPLAN
;                       HOLE-NEUE-ERZEUGER-HILF
;
; -BESTIMME-VARIABLEN-----------------------------------------------

        (COND ((NULL SITUATION)
               (MEHRFACHE-RAUS
                      VARIABLENLISTE))
              ( T  (BESTIMME-VARIABLEN
                    (CDR SITUATION)
                    (APPEND
                            (REMOVE-IF
                          #'(LAMBDA (ARGUMENT)
                              (OR
```

```
                                        (NOT (IST-VARIABLE
                                                    ARGUMENT))
                                   (NULL ARGUMENT)))

                        ; bea.: das Argument NIL wird weder als
                        ;       Variable noch als Konstante betrachtet.

                             (COND ((EQUAL (CAAR SITUATION)
                                           NEGATIONSOPERATOR)
                                    (CDADAR SITUATION))
                                   ( T  (CDAR SITUATION))) )
                       VARIABLENLISTE))) ))

; -BESTIMME-VARIABLEN-------------------------------------------------




  (DEFUN BESTIMME-VORGAENGER (AKTUELLE_GENERATION)
; ------------------------------------------------------------------
; Wert:                 Liste saemtlicher Vorgaenger-Knoten (nicht nur
;                       direkter Vorgaenger-Knoten) jedes in AKTUELLE_
;                       GENERATION angefuehrten Knotens, wobei solche
;                       Knoten, die Vorgaenger mehrerer Knoten aus
;                       AKTUELLE_GENERATION sind, jeweils nur einmal
;                       in der resultierenden Liste erscheinen.
;
; Globale Variablen:    --
;
; Eingangsparameter:      AKTUELLE_GENERATION
;                       Ein Knoten (Atom) bzw. eine Menge von Knoten
;                       (Liste), dessen bzw. deren Vorgaenger bestimmt
;                       werden sollen.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       BESTIMME-DIREKTE-VORGAENGER
;                       MEHRFACHE-RAUS
;
; aufgerufen von:       sich selbst
;                       ABHAENGIGKEITEN-AKTUALISIEREN
;                       RETTER-NACHGESCHALTET
;                       SUCHE-ALTE-RETTER
;                       SUCHE-ZERSTOERER
;
; -BESTIMME-VORGAENGER------------------------------------------------

      (COND ((NULL AKTUELLE_GENERATION) NIL)
            ( T  (LET ((VORGAENGERGENERATION
                              (BESTIMME-DIREKTE-VORGAENGER
                                    AKTUELLE_GENERATION)))
                       (MEHRFACHE-RAUS
                             (APPEND
                                 VORGAENGERGENERATION
                                 (BESTIMME-VORGAENGER
                                      VORGAENGERGENERATION))) ))) )

; -BESTIMME-VORGAENGER------------------------------------------------
```

```
  (DEFUN BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE (MERKMAL
                                                 PLANKNOTEN)
; ---------------------------------------------------------------------
; Wert:                 Anzahl instantiierter Argumente innerhalb der
;                       Vorbedingung MERKMAL der Operatorinstanz PLAN-
;                       KNOTEN.
;
;                       (Anm.: Ein Argument gilt hierbei als instanti-
;                              iert, wenn es eine Konstante (<> NIL)
;                              oder eine an eine Konstante (via Code-
;                              signation-Relation) gebundene Variable
;                              darstellt.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMAL
;                       (Vorbedingungs-)Merkmal der Operatorinstanz
;                       PLANKNOTEN, deren Anzahl instantiierter Argu-
;                       mente zu bestimmen ist.
;
;                         PLANKNOTEN
;                       Planknoten, zu dessen Vorbedingungen MERKMAL
;                       zaehlt.
;
; Nebeneffekte:         --
;
; ruft auf:             BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF
;
; aufgerufen von:       WAEHLE-UNERFUELLTES-MERKMAL
;
; -BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE----------------------------

        (BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF
                   (CDR MERKMAL)
                   PLANKNOTEN))

; -BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE----------------------------



  (DEFUN BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF (MERKMALARGUMENTE
                                                      PLANKNOTEN
                                                      &OPTIONAL
                                                           (ANZAHL 0))
; ---------------------------------------------------------------------
; Wert:                 Anzahl instantiierter Argumente innerhalb der
;                       Argumentliste MERKMALARGUMENTE.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMALARGUMENTE
;                       Liste von Argumenten einer Vorbedingung von
;                       PLANKNOTEN, unter denen die Zahl derer zu be-
;                       stimmen ist, die instantiiert sind.
;
;                         PLANKNOTEN
;                       Bezugsknoten der betrachteten Merkmalargumen-
;                       te.
;
;                         ANZAHL (optional!)
```

```
;                       Resultatwert. Anzahl der bislang als instanti-
;                       iert erkannten Argumente. Ist als optionaler
;                       Parameter definiert, um Initialisierung mit
;                       0 beim erstmaligen Aufruf von aussen einzuspa-
;                       ren.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       IST-VARIABLE
;                       KAPAARE-ORDNEN
;                       BESTIMME-ACOD-HUELLE
;
; aufgerufen von:       sich selbst
;                       BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE
;
; -BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF-----------------------

       (COND ((NULL MERKMALARGUMENTE) ANZAHL)
             ( T  (LET ((ARGUMENT
                           (CAR MERKMALARGUMENTE)))
                       (COND ((NULL ARGUMENT))
                             ((OR
                                 (NOT (IST-VARIABLE
                                             ARGUMENT))
                                 (NOT (NULL
                                         (CAR
                                            (KAPAARE-ORDNEN
                                               (BESTIMME-ACOD-HUELLE
                                                  (LIST
                                                     (LIST PLANKNOTEN
                                                           ARGUMENT))
                                               ))) )))
                              (SETQ ANZAHL
                                    (+ ANZAHL 1))) )
                       (BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF
                              (CDR MERKMALARGUMENTE)
                              PLANKNOTEN
                              ANZAHL))) ))

; -BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF-----------------------



  (DEFUN BESTIMME-ZIELKNOTEN (PLANKNOTENLISTE)
; ------------------------------------------------------------------
; Wert:                 Zielknoten des aktuell betrachteten Plans.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTENLISTE
;                       Knotenliste des aktuell betrachteten Plans.
;
; Nebeneffekte:         --
;
; ruft auf:             --
;
; aufgerufen von:       SUCHE-ALTE-ERZEUGER
;                       SUCHE-UNGELOESTE-KNOTEN
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN
;                       SUCHE-ALTE-RETTER
```

```
;                       SUCHE-ZERSTORER
;                       PLAN-AUSGEBEN
;
; -BESTIMME-ZIELKNOTEN-----------------------------------------------

       (CAR (REVERSE PLANKNOTENLISTE)))

; -BESTIMME-ZIELKNOTEN-----------------------------------------------



  (DEFUN BILDE-COD-KAPAARE (CONSTRAINTLISTE
                            PLANKNOTEN
                            ARGUMENT)
; ------------------------------------------------------------------
; Wert:                 Liste zweielementiger Unterlisten mit folgen-
;                       dem Aufbau:
;                         1. Element: Planknoten,
;                         2. Element: Argument.
;                       Saemtliche Argumente innerhalb dieser Unterli-
;                       sten sind explizit (oder direkt) codesigniert
;                       mit ARGUMENT.
;
;                       (Anm.: Die Angabe des Bezug-Planknotens er-
;                              folgt jeweils zur eindeutigen Kenn-
;                              zeichnung.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      CONSTRAINTLISTE
;                       Liste von Wertconstraints, welche den gesamten
;                       KNOTEN-CODESIGNATIONS-Eintrag der Operator-In-
;                       stanz PLANKNOTEN darstellt.
;
;                         ARGUMENT (-> PLANKNOTEN)
;                       Argument (mit Angabe des Bezugs-Knotens zur
;                       eindeutigen Kennzeichnung), dessen explizite
;                       Codesignations-Partner-Argumente zu bestimmen
;                       sind.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       ENTSPRECHENDES-ARGUMENT
;
; aufgerufen von:       sich selbst
;                       BILDE-KAPAARE
;
; -BILDE-COD-KAPAARE-------------------------------------------------

       (COND ((NULL CONSTRAINTLISTE)
              (LIST (LIST PLANKNOTEN
                          ARGUMENT)))
             ( T  (LET* ((CONSTRAINT
                                (CAR CONSTRAINTLISTE))
                         (CONSTRAINTKNOTEN
                              (WERTCONSTRAINT-KNOTEN
                                     CONSTRAINT))
                         (CONSTRAINTMERKMALE
                              (WERTCONSTRAINT-MERKMALPAAR
                                     CONSTRAINT))
```

```
                          (CONSTRAINTKNOTEN1
                               (CAR CONSTRAINTKNOTEN))
                          (CONSTRAINTKNOTEN2
                               (CADR CONSTRAINTKNOTEN))
                          (CONSTRAINTMERKMAL1
                               (CAR CONSTRAINTMERKMALE))
                          (CONSTRAINTMERKMAL2
                               (CADR CONSTRAINTMERKMALE)))
                         (APPEND
                             (COND ((AND
                                        (EQL PLANKNOTEN
                                             CONSTRAINTKNOTEN1)
                                        (MEMBER ARGUMENT
                                                CONSTRAINTMERKMAL1))
                                    (LIST
                                     (LIST CONSTRAINTKNOTEN2
                                           (ENTSPRECHENDES-ARGUMENT
                                              ARGUMENT
                                              CONSTRAINTMERKMAL1
                                              CONSTRAINTMERKMAL2))) )
                                   ((AND
                                        (EQL PLANKNOTEN
                                             CONSTRAINTKNOTEN2)
                                        (MEMBER ARGUMENT
                                                 CONSTRAINTMERKMAL2))
                                    (LIST
                                     (LIST CONSTRAINTKNOTEN1
                                           (ENTSPRECHENDES-ARGUMENT
                                              ARGUMENT
                                              CONSTRAINTMERKMAL2
                                              CONSTRAINTMERKMAL1))) ))
                             (BILDE-COD-KAPAARE
                                  (CDR CONSTRAINTLISTE)
                                  PLANKNOTEN
                                  ARGUMENT))) )))

; -BILDE-COD-KAPAARE-----------------------------------------------------



  (DEFUN BILDE-KAPAARE (CONSTRAINTLISTE
                        CONSTRAINTART
                        PLANKNOTEN
                        ARGUMENT)
; ---------------------------------------------------------------------
; Wert:                 - Falls CONSTRAINTART = 'KNOTEN-
;                                               CODESIGNATIONS:
;                         siehe Wert von BILDE-COD-KAPAARE.
;                       - Sonst:
;                         siehe Wert von BILDE-NONCOD-KAPAARE.
;
; Globale Variablen:    --
;
; Eingangsparameter:      CONSTRAINTLISTE
;                       Liste von Wertconstraints, welche den gesamten
;                       KNOTEN-(NON)CODESIGNATIONS-Eintrag der Opera-
;                       tor-Instanz PLANKNOTEN darstellt.
;
;                         CONSTRAINTART
;                       Indikator fuer die Art der in CONSTRAINTLISTE
;                       angefuehrten Wertconstraints (= 'KNOTEN-CODE-
```

```
;                       SIGNATIONS oder 'KNOTEN-NONCODESIGNATIONS).
;
;                         ARGUMENT (-> PLANKNOTEN)
;                       Argument (mit Angabe des Bezugs-Knotens zur
;                       eindeutigen Kennzeichnung), dessen explizite
;                       (Non)Codesignations-Partner-Argumente zu be-
;                       stimmen sind.
;
; Nebeneffekte:         --
;
; ruft auf:             BILDE-COD-KAPAARE
;                       BILDE-NONCOD-KAPAARE
;
; aufgerufen von:       BESTIMME-DIREKTE-ABINDUNGEN
;
; -BILDE-KAPAARE-----------------------------------------------------

       (FUNCALL
            (IF (EQUAL CONSTRAINTART
                       'KNOTEN-CODESIGNATIONS)
                'BILDE-COD-KAPAARE
                'BILDE-NONCOD-KAPAARE)
            CONSTRAINTLISTE
            PLANKNOTEN
            ARGUMENT))

; -BILDE-KAPAARE-----------------------------------------------------



  (DEFUN BILDE-NONCOD-KAPAARE (CONSTRAINTLISTE
                               PLANKNOTEN
                               ARGUMENT)
; ------------------------------------------------------------------
; Wert:                 Liste zweielementiger Unterlisten mit folgen-
;                       dem Aufbau:
;                         1. Element: Planknoten,
;                         2. Element: Argument.
;                       Saemtliche Argumente innerhalb dieser Unterli-
;                       sten sind explizit (oder direkt) noncodesig-
;                       niert mit ARGUMENT.
;
;                       (Anm.: Die Angabe des Bezug-Planknotens er-
;                              folgt jeweils zur eindeutigen Kenn-
;                              zeichnung.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      CONSTRAINTLISTE
;                       Liste von Wertconstraints, welche den gesamten
;                       KNOTEN-NONCODESIGNATIONS-Eintrag der Operator-
;                       Instanz PLANKNOTEN darstellt.
;
;                         ARGUMENT (-> PLANKNOTEN)
;                       Argument (mit Angabe des Bezugs-Knotens zur
;                       eindeutigen Kennzeichnung), dessen explizite
;                       Noncodesignations-Partner-Argumente zu bestim-
;                       men sind.
;
; Nebeneffekte:         --
;
```

```
; ruft auf:            sich selbst
;                      LOKALE-CONSTRAINTS-AUSWERTEN
;                      ANZAHL-ARGUMENTE
;                      ANZAHL-CODESIGNIERTER-ARGUMENTE
;                      ENTSPRECHENDES-ARGUMENT
;
; aufgerufen von:      sich selbst
;                      BILDE-KAPAARE
;
; -BILDE-NONCOD-KAPAARE-----------------------------------------------

       (COND ((NULL CONSTRAINTLISTE)
              (LOKALE-CONSTRAINTS-AUSWERTEN
                        PLANKNOTEN
                        ARGUMENT))
             ( T  (LET* ((CONSTRAINT
                              (CAR CONSTRAINTLISTE))
                         (CONSTRAINTKNOTEN
                              (WERTCONSTRAINT-KNOTEN
                                    CONSTRAINT))
                         (CONSTRAINTMERKMALE
                              (WERTCONSTRAINT-MERKMALPAAR
                                    CONSTRAINT))
                         (CONSTRAINTKNOTEN1
                              (CAR CONSTRAINTKNOTEN))
                         (CONSTRAINTKNOTEN2
                              (CADR CONSTRAINTKNOTEN))
                         (CONSTRAINTMERKMAL1
                              (CAR CONSTRAINTMERKMALE))
                         (CONSTRAINTMERKMAL2
                              (CADR CONSTRAINTMERKMALE))
                         (ARGUMENT_ANZAHL
                              0))
                        (APPEND
                         (COND
                           ((AND
                              (EQL PLANKNOTEN
                                   CONSTRAINTKNOTEN1)
                              (MEMBER ARGUMENT
                                      CONSTRAINTMERKMAL1)
                              (OR
                                (<=
                                  (SETQ
                                      ARGUMENT_ANZAHL
                                      (ANZAHL-ARGUMENTE
                                           CONSTRAINTMERKMAL1))
                                  1)
                                (<=
                                  (- ARGUMENT_ANZAHL
                                     (ANZAHL-CODESIGNIERTER-ARGUMENTE
                                          CONSTRAINTKNOTEN1
                                          CONSTRAINTMERKMAL1
                                          CONSTRAINTKNOTEN2
                                          CONSTRAINTMERKMAL2))
                                  1)))
                            (LIST
                                (LIST
                                    CONSTRAINTKNOTEN2
                                    (ENTSPRECHENDES-ARGUMENT
                                           ARGUMENT
                                           CONSTRAINTMERKMAL1
                                           CONSTRAINTMERKMAL2))) )
```

```
                              ((AND
                                 (EQL PLANKNOTEN
                                      CONSTRAINTKNOTEN2)
                                 (MEMBER ARGUMENT
                                         CONSTRAINTMERKMAL2)
                                 (OR
                                   (<=
                                     (SETQ
                                         ARGUMENT_ANZAHL
                                         (ANZAHL-ARGUMENTE
                                              CONSTRAINTMERKMAL2))
                                     1)
                                   (<=
                                     (- ARGUMENT_ANZAHL
                                        (ANZAHL-CODESIGNIERTER-ARGUMENTE
                                             CONSTRAINTKNOTEN1
                                             CONSTRAINTMERKMAL1
                                             CONSTRAINTKNOTEN2
                                             CONSTRAINTMERKMAL2))
                                     1)))
                               (LIST
                                   (LIST
                                      CONSTRAINTKNOTEN1
                                      (ENTSPRECHENDES-ARGUMENT
                                             ARGUMENT
                                             CONSTRAINTMERKMAL2
                                             CONSTRAINTMERKMAL1))) ))
                          (BILDE-NONCOD-KAPAARE
                                  (CDR CONSTRAINTLISTE)
                                  PLANKNOTEN
                                  ARGUMENT))) )))

; -BILDE-NONCOD-KAPAARE-----------------------------------------------



  (DEFUN CONSTRAINT-EINBAU (KNOTEN1
                            MERKMAL1
                            KNOTEN2
                            MERKMAL2
                            PLAN)
; ---------------------------------------------------------------------
; Wert:                 Neu erzeugtes Wertconstraint, das nach Aus-
;                       fuehrung von CONSTRAINT-EINBAU bereits in PLAN
;                       eingebaut ist (Nebeneffekt) und dessen Ein-
;                       traege aus den uebergebenen Argumenten be-
;                       stimmt worden sind.
;
; Globale Variablen:    --
;
; Eingangsparameter:      KNOTEN1, KNOTEN2
;                       Knoten-Paar, welches, als neu erzeugte Liste,
;                       den Eintrag WERTCONSTRAINT-KNOTEN des neu er-
;                       zeugten Wertconstraints bildet.
;
;                         MERKMAL1, MERKMAL2
;                       Merkmal-Paar, welches, als neu erzeugte Liste,
;                       den Eintrag WERTCONSTRAINT-MERKMALPAAR des neu
;                       erzeugten Wertconstraints bildet.
;
;                         PLAN
```

```
;                       Gerade bearbeiteter, zuvor neu generierter,
;                       noch nicht vollstaendig aktualisierter Plan,
;                       in dessen Feld WERTCONSTRAINTLISTE das neu er-
;                       zeugte Wertconstraint eingebaut wird.
;
; Nebeneffekte:         Erzeugen eines neuen Wertconstraints, dessen
;                       Eintraege durch die uebergebenen Argumente be-
;                       stimmt sind, und Einbauen eben dieses Con-
;                       straints in PLAN.
;
; ruft auf:             --
;
; aufgerufen von:       ERZEUGER-ERFUELLT-MERKMAL
;                       EINTRAEGE-AKTUALISIEREN-R
;                       SEPARIEREN
;
; -CONSTRAINT-EINBAU-------------------------------------------------

       (LET ((NEUES_CONSTRAINT
                  (MAKE-WERTCONSTRAINT
                       :KNOTEN
                          (LIST KNOTEN1
                                KNOTEN2)
                       :MERKMALPAAR
                          (LIST MERKMAL1
                                MERKMAL2))) )
             (SETF (PLAN-WERTCONSTRAINTLISTE PLAN)
                   (CONS
                       NEUES_CONSTRAINT
                       (PLAN-WERTCONSTRAINTLISTE PLAN)))
             NEUES_CONSTRAINT))

; -CONSTRAINT-EINBAU-------------------------------------------------



  (DEFUN CONSTRAINTS-ORDNEN (CONSTRAINTLISTE
                             PLANKNOTEN
                             MERKMAL
                             &OPTIONAL KMPAAR_LISTE
                                       CL_REST)
; ------------------------------------------------------------------
; Wert:                 Zweielementige Liste mit folgendem Aufbau:
;                         1. Element: Liste zweielementiger Unterli-
;                                     sten (= Knoten-Merkmal-Paare),
;                                   wobei hierin saemtliche Merkmale
;                                   (mit Angabe des Bezug-Plankno-
;                                   tens zur eindeutigen Kennzeich-
;                                   nung) vertreten sind, die mit
;                                   MERKMAL (-> PLANKNOTEN) explizit
;                                   (non)codesignieren.
;                         2. Element: Liste aller Wertconstraints aus
;                                   CONSTRAINTLISTE, die fuer MERK-
;                                   MAL (-> PLANKNOTEN) irrelevant
;                                   sind.
;
; Globale Variablen:    --
;
; Eingangsparameter:      CONSTRAINTLISTE
;                       Liste saemtlicher (Non)Codesignation-Wertcon-
;                       straints von PLANKNOTEN, die bezueglich der
```

```
;                       Relevanz fuer MERKMAL (-> PLANKNOTEN) zu ord-
;                       nen sind.
;
;                         PLANKNOTEN
;                       Bezugs-Planknoten von CONSTRAINTLISTE und
;                       MERKMAL.
;
;                         MERKMAL
;                       Merkmal von PLANKNOTEN, das als Ordnungskri-
;                       terium herangezogen wird.
;
;                         KMPAAR_LISTE, CL_REST (optional!)
;                       Die beiden Komponenten der Resultatliste. Sind
;                       als optionale Parameter definiert, um Ini-
;                       tialisierung mit NIL beim erstmaligen Aufruf
;                       von aussen einzusparen.
;
; Nebeneffekte:         Erzeugen der Resultatliste, wobei deren beide
;                       Teillisten ebenfalls neu zusammengesetzt wer-
;                       den.
;
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       MERKMALBINDUNGEN-AUSGEBEN
;
; -CONSTRAINTS-ORDNEN-------------------------------------------------

       (COND ((NULL CONSTRAINTLISTE)
              (LIST KMPAAR_LISTE
                    CL_REST))
             ( T  (LET* ((CONSTRAINT
                                (CAR CONSTRAINTLISTE))
                         (CONSTRAINTKNOTEN
                              (WERTCONSTRAINT-KNOTEN
                                       CONSTRAINT))
                         (CONSTRAINTMERKMALE
                              (WERTCONSTRAINT-MERKMALPAAR
                                       CONSTRAINT))
                         (CK1
                              (CAR CONSTRAINTKNOTEN))
                         (CK2
                              (CADR CONSTRAINTKNOTEN))
                         (CM1
                              (CAR CONSTRAINTMERKMALE))
                         (CM2
                              (CADR CONSTRAINTMERKMALE)))
                        (COND ((AND
                                   (EQL PLANKNOTEN
                                        CK1)
                                   (OR
                                      (EQL MERKMAL
                                           CM1)
                                      (EQL (CADR MERKMAL)
                                           CM1)))
                               (SETQ KMPAAR_LISTE
                                     (CONS
                                         (LIST CK2
                                               CM2)
                                        KMPAAR_LISTE)))
                              ((AND
                                   (EQL PLANKNOTEN
```

```
                                        CK2)
                                    (OR
                                       (EQL MERKMAL
                                            CM2)
                                       (EQL (CADR MERKMAL)
                                            CM2)))
                                  (SETQ KMPAAR_LISTE
                                        (CONS
                                            (LIST CK1
                                                  CM1)
                                            KMPAAR_LISTE)))
                                 ( T  (SETQ CL_REST
                                            (CONS
                                                CONSTRAINT
                                                CL_REST))) )
                           (CONSTRAINTS-ORDNEN
                                    (CDR CONSTRAINTLISTE)
                                    PLANKNOTEN
                                    MERKMAL
                                    KMPAAR_LISTE
                                    CL_REST))) ))

; -CONSTRAINTS-ORDNEN-------------------------------------------------



  (DEFUN EINTRAEGE-AKTUALISIEREN-E (KNOTENLISTE
                                    NEU
                                    CONSTRAINT
                                    NUTZER
                                    ERZEUGER
                                    EFFEKT)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      KNOTENLISTE
;                       Knotenliste des gerade bearbeiteten Plans.
;
;                         NEU
;                       Indikator dafuer, dass ein neuer Erzeuger-Kno-
;                       ten eingebaut wurde (dann Wert: T, sonst: NIL)
;
;                         CONSTRAINT
;                       Neu erzeugtes und bereits im gerade bearbeite-
;                       ten Plan eingebautes Codesignation-Wertcon-
;                       straint zwischen Nutzer- und Erzeuger-Knoten.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des gerade in
;                       Bearbeitung befindlichen Plans.
;
;                         ERZEUGER
;                       Zur Erfuellung der gewaehlten unerfuellten
;                       Vorbedingung von NUTZER gewaehlter Knoten des
;                       gerade bearbeiteten Plans.
;
;                         EFFEKT
;                       Nichtnegiertes Nachbedingungs-Merkmal von ER-
;                       ZEUGER, das mit dem gewaehlten unerfuellten
```

```
;                       Vorbedingungs-Merkmal von NUTZER codesignier-
;                       bar ist und seiner Erfuellung dient.
;
; Nebeneffekte:         - Falls ein neuer Erzeuger-Knoten eingebaut
;                         wurde:
;                         Initialisieren des Vorgaenger-Eintrags des
;                         ERZEUGERs mit einer nur aus dem Startknoten
;                         des gerade bearbeiteten Plans bestehenden
;                         Liste.
;                       - Verbinden von NUTZER und ERZEUGER mittels
;                         des neu erzeugten Wertconstraints CONSTRAINT
;                         dadurch, dass die jeweiligen CODESIGNATIONS-
;                         Eintraege um CONSTRAINT erweitert werden.
;                       - Entsprechende Eintragung in das Merkmal-Er-
;                         zeugungsfeld von ERZEUGER.
;                       - Falls der gewaehlte Erzeuger-Knoten noch
;                         nicht in Vorgaenger-Beziehung zum Nutzer-
;                         Knoten steht:
;                         Erweitern der Vorgaenger-Liste von NUTZER um
;                         ERZEUGER mit anschliessender Zeitconstraint-
;                         netz-Optimierung.
;
; ruft auf:             IST-VORGAENGER
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN
;                       BESTIMME-STARTKNOTEN
;
; aufgerufen von:       ERZEUGER-ERFUELLT-MERKMAL
;
; -EINTRAEGE-AKTUALISIEREN-E----------------------------------------

       (COND (NEU
                (SETF (KNOTEN-VORGAENGER ERZEUGER)
                    (LIST
                        (BESTIMME-STARTKNOTEN
                                KNOTENLISTE))) ))
       (SETF (KNOTEN-CODESIGNATIONS NUTZER)
             (CONS
                 CONSTRAINT
                 (KNOTEN-CODESIGNATIONS NUTZER)))
       (SETF (KNOTEN-CODESIGNATIONS ERZEUGER)
             (CONS
                 CONSTRAINT
                 (KNOTEN-CODESIGNATIONS ERZEUGER)))
       (SETF (KNOTEN-erzeugt_fuer ERZEUGER)
             (CONS
                 (LIST NUTZER
                       EFFEKT)
                 (KNOTEN-erzeugt_fuer ERZEUGER)))
       (COND ((NOT (IST-VORGAENGER
                          ERZEUGER
                          NUTZER))
              (SETF (KNOTEN-VORGAENGER NUTZER)
                    (CONS
                        ERZEUGER
                        (KNOTEN-VORGAENGER NUTZER)))
              (ZEITCONSTRAINTNETZ-OPTIMIEREN
                     KNOTENLISTE
                     NUTZER
                     ERZEUGER))) )

; -EINTRAEGE-AKTUALISIEREN-E----------------------------------------
```

```
  (DEFUN ENTSPRECHENDES-ARGUMENT (ARGUMENT
                                  MERKMAL1
                                  MERKMAL2)
; ---------------------------------------------------------------------
; Wert:                 Argument innerhalb von MERKMAL2, das bezueg-
;                       lich seiner Position dem uebergebenen ARGUMENT
;                       aus MERKMAL1 entspricht.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ARGUMENT
;                       Argument innerhalb von MERKMAL1, zu dem das
;                       positionsmaessig entsprechende innerhalb von
;                       MERKMAL2 bestimmt werden soll.
;
;                         MERKMAL1
;                       Merkmal, innerhalb dem das uebergebene ARGU-
;                       MENT enthalten ist.
;
;                         MERKMAL2
;                       Merkmal, aus dem das entsprechende Argument
;                       bestimmt werden soll.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       BILDE-COD-KAPAARE
;                       BILDE-NONCOD-KAPAARE
;
; -ENTSPRECHENDES-ARGUMENT-------------------------------------------

        (COND ((EQUAL ARGUMENT
                      (CAR MERKMAL1))
               (CAR MERKMAL2))
              ( T  (ENTSPRECHENDES-ARGUMENT
                          ARGUMENT
                          (CDR MERKMAL1)
                          (CDR MERKMAL2))) ))

; -ENTSPRECHENDES-ARGUMENT-------------------------------------------



  (DEFUN ERMITTLE-ERZEUGER (PLANKNOTENLISTE
                            NUTZER
                            MERKMAL)
; ---------------------------------------------------------------------
; Wert:                 - Falls Erzeuger fuer MERKMAL (-> NUTZER) exi-
;                         stent:
;                         Liste dreielementiger Unterlisten mit fol-
;                         gendem Aufbau:
;                         1. Element: Kennung
;                                     (= 'ALT, falls Erzeuger bereits
;                                          im Plan eingebaut ist;
;                                    = 'NEU, falls Erzeuger noch
;                                          neu, also noch nicht im
```

```
;                                        Plan eingebaut ist;
;                                   = 'NOCH_NICHT_ETABLIERT, falls
;                                        Erzeuger vorgemerkt,
;                                        d.h. bereits im Plan
;                                        eingebaut, aber wegen
;                                        Zerstoerer noch nicht
;                                        bzw. nicht mehr als Er-
;                                        zeuger etabliert ist.),
;                           2. Element: Erzeuger(-Knoten), welcher fuer
;                                    den Nutzer(-Knoten) die noch un-
;                                    erfuellte Vorbedingung MERKMAL
;                                    erzeugen koennte,
;                           3. Element: Effekt
;                                    (Nachbedingung des Erzeugers,
;                                    welche mit der Vorbedingung
;                                    MERKMAL des Nutzers codesignier-
;                                    bar ist).
;                         - Sonst:
;                           NIL.
;
; Globale Variablen:      VERFUEGBARE_OPERATOREN
;
; Eingangsparameter:        PLANKNOTENLISTE
;                         Knotenliste des aktuell betrachteten Plans.
;
;                           NUTZER
;                         Ungeloester Knoten innerhalb des aktuell be-
;                         trachteten Plans, fuer dessen noch unerfuell-
;                         te Vorbedingung MERKMAL nach Erzeugern ge-
;                         sucht wird.
;
;                           MERKMAL
;                         Unerfuelltes (Vorbedingungs-)Merkmal von NUT-
;                         ZER, fuer dessen Erfuellung nach Erzeugern
;                         gesucht wird.
;
; Nebeneffekte:           Erzeugen der Resultatliste.
;
; ruft auf:               ERZEUGER-VORGEMERKT
;                         SUCHE-ALTE-ERZEUGER
;                         HOLE-NEUE-ERZEUGER
;
; aufgerufen von:         PLAN-ERWEITERUNG
;
; -ERMITTLE-ERZEUGER-------------------------------------------------

       (COND ((ERZEUGER-VORGEMERKT
                    (KNOTEN-ist_abhaengig_von
                                 NUTZER)
                    MERKMAL))
             ( T  (APPEND
                      (SUCHE-ALTE-ERZEUGER
                              PLANKNOTENLISTE
                              NUTZER
                              MERKMAL)
                      (HOLE-NEUE-ERZEUGER
                              VERFUEGBARE_OPERATOREN
                              MERKMAL))) ))

; -ERMITTLE-ERZEUGER-------------------------------------------------
```

```
  (DEFUN ERZEUGE-INITIALPLAN ()
; ---------------------------------------------------------------------
; Wert:                 Initialplan, dessen Knotenliste lediglich aus
;                       den beiden Operator-Instanzen Start- und Ziel-
;                       knoten besteht.
;                       Die Eintraege des Startknotens sind mit NIL
;                       initialisiert bis auf folgende Ausnahmen:
;                       - Der Name wird vom LISP-System generiert.
;                       - Als Variablen werden die in der Startsitua-
;                         tion vorkommenden Variablen eingetragen.
;                       - Als Operator wird 'START festgelegt.
;                       Die Eintraege vom Zielknoten sind ebenfalls
;                       mit NIL initialisiert bis auf folgende Ausnah-
;                       men:
;                       - Der Name wird vom LISP-System generiert.
;                       - Als Variablen werden die in der Finalsitua-
;                         tion vorkommenden Variablen eingetragen.
;                       - Als Operator wird 'ZIEL festgelegt.
;                       - Eine erste Operatorordnung wird dadurch
;                         festgelegt, dass eine nur aus dem Startkno-
;                         ten bestehende Liste als Vorgaenger des
;                         Zielknotens etabliert wird.
;                       Die Wertconstraintliste des Initialplans ist
;                       leer.
;
; Globale Variablen:    START (= Operator des Startknotens)
;                       ZIEL  (= Operator des Zielknotens)
;
; Eingangsparameter:    --
;
; Nebeneffekte:         Erzeugen zweier neuer Knotenstrukturen:
;                         Start- und Zielknoten.
;                       Erzeugen eines Initialplans bestehend aus
;                       Start- und Zielknoten.
;
; ruft auf:             BESTIMME-VARIABLEN
;
; aufgerufen von:       FKNLP
;
; -ERZEUGE-INITIALPLAN-----------------------------------------------

       (LET* ((STARTKNOTEN
                  (MAKE-KNOTEN
                        :VARIABLEN
                              (BESTIMME-VARIABLEN
                                    (OPERATOR-NACHBEDINGUNGEN
                                                START))
                        :OPERATOR
                              'START))
              (ZIELKNOTEN
                  (MAKE-KNOTEN
                        :VARIABLEN
                              (BESTIMME-VARIABLEN
                                    (OPERATOR-VORBEDINGUNGEN
                                                ZIEL))
                        :OPERATOR
                              'ZIEL
                        :VORGAENGER
                              (LIST STARTKNOTEN))) )
             (MAKE-PLAN
```

```
                  :KNOTENLISTE
                         (LIST STARTKNOTEN
                               ZIELKNOTEN))) )

; -ERZEUGE-INITIALPLAN-----------------------------------------------



  (DEFUN ERZEUGER-ETABLIEREN (NUTZER
                              MERKMAL
                              ERZEUGER)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      NUTZER
;                       Gewaehlter ungeloester Knoten im gerade bear-
;                       beiteten Plan.
;
;                         MERKMAL
;                       Noch unerfuellte Vorbedingung von NUTZER, fuer
;                       die nun ERZEUGER etabliert wird.
;
;                         ERZEUGER
;                       Die Vorbedingung MERKMAL (-> NUTZER) erzeugen-
;                       der Knoten des gerade bearbeiteten Plans, der
;                       als solcher nun beim NUTZER etabliert werden
;                       soll.
;
; Nebeneffekte:         Eintragen von ERZEUGER als etablierter Erzeu-
;                       ger in der Abhaengigkeitsliste von NUTZER.
;                       Dazu wird aus ERZEUGER und MERKMAL eine zwei-
;                       elementige Liste (= Knoten-Merkmal-Paar) gene-
;                       riert und zu den bisherigen Abhaengigkeitsein-
;                       traegen von NUTZER hinzugefuegt. Falls ERZEU-
;                       GER schon vorgemerkt, also als solcher erkannt
;                       und verwaltet, aber beim Nutzer-Knoten noch
;                       nicht etabliert war, wird diese Vormerkung in
;                       Abhaengigkeitsliste von NUTZER geloescht und
;                       durch den neuen Eintrag (s.o.) ersetzt.
;
; ruft auf:             ERZEUGER-VORGEMERKT
;
; aufgerufen von:       KONFLIKT-LOESUNGEN
;                       NUTZER-ABHAENGIGKEIT-EINTRAGEN
;
; -ERZEUGER-ETABLIEREN-----------------------------------------------

        (LET ((ABHAENGIGKEITEN
                      (KNOTEN-ist_abhaengig_von NUTZER)))
             (SETF (KNOTEN-ist_abhaengig_von NUTZER)
                   (CONS
                       (LIST ERZEUGER
                             MERKMAL)
                       (REMOVE
                            (CAR (ERZEUGER-VORGEMERKT
                                        ABHAENGIGKEITEN
                                        MERKMAL))
                           ABHAENGIGKEITEN
                           :TEST 'EQUAL))) ))
```

```
; -ERZEUGER-ETABLIEREN-----------------------------------------------



  (DEFUN FERTIG (AKTUELLER_PLAN)
; ---------------------------------------------------------------------
; Wert:                 - T, falls saemtliche Knoten von AKTUELLER_
;                         PLAN geloest, d.h. Abhaengigkeiten fuer alle
;                         Vorbedingungen der entpsrechenden Operator-
;                         Strukturen etabliert sind.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      AKTUELLER_PLAN
;                       Aktuell betrachteter (unvollstaendiger) nicht-
;                       linearer Plan, der das erste Element (von
;                       links) der Plan-Warteschlange, die in PLANEN
;                       verwaltet wird, bildet und dessen Knotenliste
;                       auf Geloestheit zu untersuchen ist.
;
; Nebeneffekte:         --
;
; ruft auf:             GELOEST
;
; aufgerufen von:       PLANEN
;
; -FERTIG-----------------------------------------------------------

       (GELOEST
             (PLAN-KNOTENLISTE
                    AKTUELLER_PLAN)))

; -FERTIG-----------------------------------------------------------



  (DEFUN GELOEST (PLANKNOTEN)
; ---------------------------------------------------------------------
; Wert:                 - T, falls die Operator-Instanz PLANKNOTEN bzw.
;                         saemtliche Operatorinstanzen aus der Liste
;                         PLANKNOTEN geloest, d.h. fuer alle Vorbedin-
;                         gungen der entsprechenden Operator-Struk-
;                         tur(en) Abhaengigkeiten etabliert sind.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTEN
;                       (Atomarer) Planknoten/Liste von Planknoten,
;                       der/die auf Geloestheit zu untersuchen ist/
;                       sind.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
;
; aufgerufen von:       sich selbst
;                       FERTIG
```

```
;                       SUCHE-UNGELOESTE-KNOTEN-HILF
;
; -GELOEST-----------------------------------------------------------

        (COND ((NULL PLANKNOTEN) T)
              ((ATOM PLANKNOTEN)
               (= (LENGTH
                        (BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
                                PLANKNOTEN))
                  (LENGTH
                        (OPERATOR-VORBEDINGUNGEN
                                (EVAL (KNOTEN-OPERATOR
                                            PLANKNOTEN))) )))
              ( T  (AND
                      (GELOEST
                          (CAR PLANKNOTEN))
                      (GELOEST
                          (CDR PLANKNOTEN))) )))

; -GELOEST-----------------------------------------------------------



  (DEFUN HOLE-NEUE-ERZEUGER (OPERATORNAMEN_LISTE
                             MERKMAL)
; ------------------------------------------------------------------
; Wert:                 - Falls Erzeuger in OPERATORNAMEN_LISTE exi-
;                         stent:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung (= 'NEU),
;                             2. Element: Erzeuger(-Knoten), welcher
;                                     fuer den Nutzer(-Knoten) die
;                                     noch unerfuellte Vorbedin-
;                                     gung MERKMAL erzeugen koenn-
;                                     te, dessen Eintraege aber
;                                     bis auf KNOTEN-NAME, -VA-
;                                     RIABLEN und -OPERATOR noch
;                                     nicht aktualisiert, sondern
;                                     erst noch mit NIL initiali-
;                                     siert sind,
;                             3. Element: Effekt (Nachbedingung des
;                                     Erzeugers, welche mit der
;                                     Vorbedingung MERKMAL des
;                                     Nutzers codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      OPERATORNAMEN_LISTE
;                       Liste von Operator-Namen, an die die noch zur
;                       Verfuegung stehenden Operator-Strukturen als
;                       Werte gebunden sind.
;
;                         MERKMAL
;                       Gewaehlte unerfuellte Vorbedingung des im ak-
;                       tuell betrachteten Plan gewaehlten ungeloesten
;                       Knotens.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
```

```
;
; ruft auf:             sich selbst
;                       BESTIMME-NICHTNEGIERTE-EFFEKTE
;                       HOLE-NEUE-ERZEUGER-HILF
;
; aufgerufen von:       sich selbst
;                       ERMITTLE-ERZEUGER
;                       ERMITTLE-RETTER
;
; -HOLE-NEUE-ERZEUGER----------------------------------------------

       (COND ((NULL OPERATORNAMEN_LISTE) NIL)
             ( T  (LET* ((OPERATORNAME
                              (CAR OPERATORNAMEN_LISTE))
                         (EFFEKTE
                              (BESTIMME-NICHTNEGIERTE-EFFEKTE
                                    OPERATORNAME)))
                        (APPEND
                            (HOLE-NEUE-ERZEUGER-HILF
                                    OPERATORNAME
                                    EFFEKTE
                                    MERKMAL)
                            (HOLE-NEUE-ERZEUGER
                                    (CDR OPERATORNAMEN_LISTE)
                                    MERKMAL))) )))

; -HOLE-NEUE-ERZEUGER----------------------------------------------



  (DEFUN HOLE-NEUE-ERZEUGER-HILF (OPERATORNAME
                                  EFFEKTE
                                  MERKMAL)
; ---------------------------------------------------------------------
; Wert:                 - Falls die an OPERATORNAME als Wert gebundene
;                         Operator-Struktur in der Lage ist, die uner-
;                         fuellte Vorbedingung MERKMAL zu erzeugen:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung (= 'NEU),
;                             2. Element: Erzeuger(-Knoten), welcher
;                                         fuer den Nutzer(-Knoten) die
;                                         noch unerfuellte Vorbedin-
;                                         gung MERKMAL erzeugen koenn-
;                                         te, dessen Eintraege aber
;                                         bis auf KNOTEN-NAME, -VA-
;                                         RIABLEN und -OPERATOR (=
;                                         (eval OPERATORNAME) ) noch
;                                         nicht aktualisiert, sondern
;                                         erst noch mit NIL initiali-
;                                         siert sind,
;                             3. Element: Effekt (Nachbedingung des
;                                         Erzeugers, welche mit der
;                                         Vorbedingung MERKMAL des
;                                         Nutzers codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      OPERATORNAME
```

```
;                       Operator-Name, an den eine der verfuegbaren
;                       Operator-Strukturen, welche eventuell als Er-
;                       zeuger fuer MERKMAL fungieren koennte, als
;                       Wert gebunden ist.
;
;                         EFFEKTE
;                       Liste saemtlicher nichtnegierter Nachbedin-
;                       gungs-Merkmale der an OPERATORNAME als Wert
;                       gebundenen Operator-Struktur.
;
;                         MERKMAL
;                       Gewaehlte unerfuellte Vorbedingung des im ak-
;                       tuell betrachteten Plan gewaehlten ungeloesten
;                       Knotens.
;
; Nebeneffekte:         Erzeugen einer neuen Knoten-Struktur (oder so-
;                       gar mehrerer davon; aber dann alle mit demsel-
;                       ben Knoten-Operator OPERATORNAME), falls die
;                       an OPERATORNAME als Wert gebundene Operator-
;                       Struktur als neuer Erzeuger (vielleicht sogar
;                       in mehrfacher Weise; dann mittels verschiede-
;                       ner Effekte) fungieren kann.
;                       Erzeugen der Resultatliste.
;
; ruft auf:             sich selbst
;                       MATCH-MOEGLICH
;                       BESTIMME-VARIABLEN
;
; aufgerufen von:       sich selbst
;                       HOLE-NEUE-ERZEUGER
;
; -HOLE-NEUE-ERZEUGER-HILF-----------------------------------------

       (COND ((NULL EFFEKTE) NIL)
             ( T  (LET ((EFFEKT
                             (CAR EFFEKTE))
                         (OPERATORSTRUKTUR
                             (EVAL OPERATORNAME)))
                        (APPEND
                            (COND ((MATCH-MOEGLICH
                                        EFFEKT
                                        MERKMAL)
                                   (LIST
                                     (LIST
                                       'NEU
                                       (MAKE-KNOTEN
                                        :VARIABLEN
                                          (BESTIMME-VARIABLEN
                                            (APPEND
                                              (OPERATOR-VORBEDINGUNGEN
                                                    OPERATORSTRUKTUR)
                                              (OPERATOR-NACHBEDINGUNGEN
                                                    OPERATORSTRUKTUR)))
                                        :OPERATOR
                                         OPERATORNAME)
                                       EFFEKT))) )
                                  (HOLE-NEUE-ERZEUGER-HILF
                                         OPERATORNAME
                                         (CDR EFFEKTE)
                                         MERKMAL))) )))

; -HOLE-NEUE-ERZEUGER-HILF-----------------------------------------
```

```
  (DEFUN IST-ERZEUGER (MOEGLICHER_ERZEUGER
                       NUTZER
                       MERKMAL)
; ---------------------------------------------------------------------
; Wert:                 - Falls der Knoten MOEGLICHER_ERZEUGER fuer
;                         den Knoten NUTZER die Vorbedingung  MERKMAL
;                         zu erzeugen im Stande ist:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung
;                                         (= 'ALT),
;                             2. Element: Erzeuger
;                                         (= MOEGLICHER_ERZEUGER),
;                             3. Element: Effekt
;                                         (= Nachbedingungs-Merkmal
;                                            von MOEGLICHER_ERZEUGER,
;                                            welches mit der Vorbedin-
;                                            gung MERKMAL des Nutzers
;                                            codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MOEGLICHER_ERZEUGER
;                       Knoten des aktuell betrachteten Plans, der da-
;                       raufhin zu untersuchen ist, ob er die uner-
;                       fuellte Vorbedingung MERKMAL (-> NUTZER) zu
;                       erzeugen im Stande ist.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des aktuell be-
;                       trachteten Plans, fuer den ein Erzeuger be-
;                       stimmt werden soll.
;
;                         MERKMAL
;                       Gewaehlte unerfuellte Vorbedingung von NUTZER,
;                       fuer die ein Erzeuger bestimmt werden soll.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             IST-ERZEUGER-HILF
;                       BESTIMME-NICHTNEGIERTE-EFFEKTE
;
; aufgerufen von:       SUCHE-ALTE-ERZEUGER-HILF
;                       SUCHE-ALTE-RETTER-HILF
;
; -IST-ERZEUGER-------------------------------------------------------

        (IST-ERZEUGER-HILF
                MOEGLICHER_ERZEUGER
                (BESTIMME-NICHTNEGIERTE-EFFEKTE
                        (KNOTEN-OPERATOR
                                MOEGLICHER_ERZEUGER))
                NUTZER
                MERKMAL))

; -IST-ERZEUGER-------------------------------------------------------
```

```
  (DEFUN IST-ERZEUGER-HILF (MOEGLICHER_ERZEUGER
                            ERZEUGER_EFFEKTE
                            NUTZER
                            MERKMAL
                            &OPTIONAL K_E_M_TRIPELLISTE)
; ---------------------------------------------------------------------
; Wert:                 - Falls der Knoten MOEGLICHER_ERZEUGER fuer
;                         den Knoten NUTZER die Vorbedingung  MERKMAL
;                         zu erzeugen im Stande ist:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung
;                                         (= 'ALT),
;                             2. Element: Erzeuger
;                                         (= MOEGLICHER_ERZEUGER),
;                             3. Element: Effekt
;                                         (= Nachbedingungs-Merkmal
;                                            von MOEGLICHER_ERZEUGER,
;                                            welches mit der Vorbedin-
;                                            gung MERKMAL des Nutzers
;                                            codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MOEGLICHER_ERZEUGER
;                       Knoten des aktuell betrachteten Plans, der da-
;                       raufhin zu untersuchen ist, ob er die uner-
;                       fuellte Vorbedingung MERKMAL (-> NUTZER) zu
;                       erzeugen im Stande ist.
;
;                         ERZEUGER_EFFEKTE
;                       Saemtliche nichtnegierte Nachbedingungs-Merk-
;                       male des dem Knoten MOEGLICHER_ERZEUGER zuge-
;                       ordneten Operators.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des aktuell be-
;                       trachteten Plans, fuer den ein Erzeuger be-
;                       stimmt werden soll.
;
;                         MERKMAL
;                       Gewaehlte unerfuellte Vorbedingung von NUTZER,
;                       fuer die ein Erzeuger bestimmt werden soll.
;
;                         K_E_M_TRIPELLISTE (optional!)
;                       Resultatliste mit obig erwaehntem Aufbau, de-
;                       finiert als optionaler Parameter, um eine ex-
;                       plizite Initialisierung mit NIL beim erstmali-
;                       gen Aufruf von aussen mit derselben Wirkung
;                       einzusparen.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             sich selbst
;                       MERKMALE-CODESIGNIERBAR
;
```

```
; aufgerufen von:       sich selbst
;                       IST-ERZEUGER
;
; -IST-ERZEUGER-HILF-------------------------------------------------

        (COND ((NULL ERZEUGER_EFFEKTE) K_E_M_TRIPELLISTE)
              ( T  (IST-ERZEUGER-HILF
                          MOEGLICHER_ERZEUGER
                          (CDR ERZEUGER_EFFEKTE)
                          NUTZER
                          MERKMAL
                          (NCONC
                              (MERKMALE-CODESIGNIERBAR
                                    MOEGLICHER_ERZEUGER
                                    (CAR ERZEUGER_EFFEKTE)
                                    NUTZER
                                    MERKMAL)
                              K_E_M_TRIPELLISTE))) ))

; -IST-ERZEUGER-HILF-------------------------------------------------




  (DEFUN IST-NEGIERT (MERKMAL)
; --------------------------------------------------------------------
; Wert:                 - T, falls die Nachbedingung MERKMAL negiert
;                         ist.
;                       - NIL, sonst.
;
; Globale Variablen:    NEGATIONSOPERATOR
;
; Eingangsparameter:      MERKMAL
;                       (Nachbedingungs-)Merkmal, das auf Negiertheit
;                       zu untersuchen ist.
;
; Nebeneffekte:         --
;
; ruft auf:             --
;
; aufgerufen von:       BESTIMME-MERKMALLAENGE
;                       BESTIMME-NEGIERTE-EFFEKTE
;                       BESTIMME-NICHTNEGIERTE-EFFEKTE
;
; -IST-NEGIERT-------------------------------------------------------

        (EQUAL (CAR MERKMAL)
               NEGATIONSOPERATOR))

; -IST-NEGIERT-------------------------------------------------------




  (DEFUN IST-VARIABLE (VAR_ODER_CONST)
; --------------------------------------------------------------------
; Wert:                 - T, falls VAR_ODER_CONST eine Variable dar-
;                         stellt.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
```

```
; Eingangsparameter:      VAR_ODER_CONST
;                       Symbol, das daraufhin zu untersuchen ist, ob
;                       es eine Variable darstellt.
;
;                       (Anm.: Symbole mit vorangestelltem Fragezei-
;                              chen repraesentieren Variablen.)
;
; Nebeneffekte:         --
;
; ruft auf:             --
;
; aufgerufen von:       ARGUMENTE-CODESIGNIERT
;                       BESTIMME-VARIABLEN
;                       BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF
;                       KAPAARE-ORDNEN
;                       MATCH-MOEGLICH
;                       MERKMALE-CODESIGNIERBAR-HILF
;                       KAPAARE-AUSGEBEN
;
; -IST-VARIABLE-----------------------------------------------------

        (EQUAL (AREF (SYMBOL-NAME
                        VAR_ODER_CONST)
                     0)
               '#\?))

                  ; Variablen sind an vorangestelltem Frage-
                  ; zeichen zu erkennen, z.B. ?X.

; -IST-VARIABLE-----------------------------------------------------



  (DEFUN IST-VORGAENGER (KNOTEN1
                         KNOTEN2
                         &OPTIONAL VORGAENGERMENGE2)
; ------------------------------------------------------------------
; Wert:                 - Liste von Knoten (also Wert <> NIL) begin-
;                         nend mit KNOTEN1, falls KNOTEN1 Vorgaenger
;                         von KNOTEN2 ist.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      KNOTEN1
;                       Knoten, der daraufhin zu untersuchen ist, ob
;                       er in Vorgaengerbeziehung zu KNOTEN2 steht.
;
;                         KNOTEN2
;                       Knoten, bezueglich dem KNOTEN1 auf Vorgaenger-
;                       beziehung untersucht werden soll.
;
;                         VORGAENGERMENGE2 (optional!)
;                       Saemtliche Vorgaengerknoten von KNOTEN2.
;
;                       (Anm.: Um wiederholtermassen verschiedene Kno-
;                              ten KNOTEN1 auf Vorgaengerbeziehung zum
;                              gleichen KNOTEN2 effizienter zu unter-
;                              suchen, kann zusaetzlich VORGAENGERMEN-
;                              GE2 beim Aufruf von IST-VORGAENGER an-
;                              gegeben werden.)
```

```
;
; Nebeneffekte:         --
;
; ruft auf:             BESTIMME-DIREKTE-VORGAENGER
;                       IST-VORGAENGER-HILF
;
; aufgerufen von:       ABHAENGIGKEITEN-AKTUALISIEREN
;                       ABHAENGIGKEITEN-PRUEFEN
;                       EINTRAEGE-AKTUALISIEREN-E
;                       SUCHE-ALTE-ERZEUGER-HILF
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF
;                       EINTRAEGE-AKTUALISIEREN-R
;                       KONFLIKT-ANALYSIEREN
;                       RETTER-NACHGESCHALTET-HILF
;                       SUCHE-ALTE-RETTER-HILF
;                       SUCHE-ZERSTOERER-HILF
;
; -IST-VORGAENGER-----------------------------------------------------

        (COND ((NULL VORGAENGERMENGE2)
               (LET ((VORGAENGERGENERATION
                           (BESTIMME-DIREKTE-VORGAENGER
                                   KNOTEN2)))
                    (IST-VORGAENGER-HILF
                           KNOTEN1
                           VORGAENGERGENERATION)))
              ( T  (MEMBER KNOTEN1
                           VORGAENGERMENGE2))) )

; -IST-VORGAENGER-----------------------------------------------------



  (DEFUN IST-VORGAENGER-HILF (MOEGLICHER_VORGAENGERKNOTEN
                              VORGAENGERGENERATION)
; ------------------------------------------------------------------
; Wert:                 - Liste von Knoten (also Wert <> NIL) begin-
;                         nend mit MOEGLICHER_VORGAENGERKNOTEN, falls
;                         eben dieser Knoten in VORGAENGERGENERATION
;                         oder einer Vorgaengergeneration derselben
;                         vorkommt.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MOEGLICHER_VORGAENGERKNOTEN
;                       Knoten, der daraufhin zu untersuchen ist, ob
;                       er in VORGAENGERGENERATION oder in einer Vor-
;                       gaengergeneration derselben vorkommt.
;
;                         VORGAENGERGENERATION
;                       Liste von Knoten, die eine Vorgaengergenera-
;                       tion des Knotens im aktuell betrachteten Plan
;                       bilden, bezueglich dem der andere Knoten
;                       MOEGLICHER_VORGAENGERKNOTEN auf Vorgaengerbe-
;                       ziehung untersucht werden soll.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
```

```
;                       BESTIMME-DIREKTE-VORGAENGER
;
; aufgerufen von:       sich selbst
;                       IST-VORGAENGER
;
; -IST-VORGAENGER-HILF-----------------------------------------------

        (COND ((NULL VORGAENGERGENERATION) NIL)
              ( T  (OR
                      (MEMBER  MOEGLICHER_VORGAENGERKNOTEN
                               VORGAENGERGENERATION)
                      (IST-VORGAENGER-HILF
                               MOEGLICHER_VORGAENGERKNOTEN
                               (BESTIMME-DIREKTE-VORGAENGER
                                          VORGAENGERGENERATION))) )))

; -IST-VORGAENGER-HILF-----------------------------------------------



  (DEFUN KAPAARE-ORDNEN (KAPAAR_LISTE
                         &OPTIONAL KKPAAR_LISTE
                                   KVPAAR_LISTE)
; ------------------------------------------------------------------
; Wert:                 Zweielementige Liste mit folgendem Aufbau:
;                        1. Element: Liste zweielementiger Unterlisten
;                                    (= Knoten-Konstanten-Paare),
;                        2. Element: Liste zweielementiger Unterlisten
;                                    (= Knoten-Variablen-Paare).
;
; Globale Variablen:    --
;
; Eingangsparameter:      KAPAAR_LISTE
;                       Liste zweielementiger Unterlisten (= Knoten-
;                       Argument-Paare), die in Knoten-Konstanten- und
;                       Knoten-Variablen-Paare aufzuspalten sind.
;
;                         KKPAAR_LISTE (optional!)
;                       Liste zweielementiger Unterlisten (= Knoten-
;                       Konstanten-Paare), die den CAR-Teil der Resul-
;                       tatliste bildet und saemtliche Knoten-Konstan-
;                       ten-Paare aus KAPAAR_LISTE enthaelt.
;
;                         KVPAAR_LISTE (optional!)
;                       Liste zweielementiger Unterlisten (= Knoten-
;                       Variablen-Paare), die den CADR-Teil der Resul-
;                       tatliste bildet und saemtliche Knoten-Variab-
;                       len-Paare aus KAPAAR_LISTE enthaelt.
;                       KKPAAR_LISTE und KVPAAR_LISTE sind als optio-
;                       nale Parameter definiert, um Initialisierung
;                       mit NIL beim erstmaligen Aufruf von aussen
;                       einsparen zu koennen.
;
; Nebeneffekte:         Erzeugen der beiden Teillisten der Resultat-
;                       liste sowie der Resultatliste selbst.
;
; ruft auf:             sich selbst
;                       IST-VARIABLE
;
; aufgerufen von:       sich selbst
;                       ARGUMENTE-NONCODESIGNIERT
```

```
;                       BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE-HILF
;
; -KAPAARE-ORDNEN-----------------------------------------------------

        (COND ((NULL KAPAAR_LISTE)
               (LIST KKPAAR_LISTE
                     KVPAAR_LISTE))
              ( T  (LET ((KAPAAR    (CAR KAPAAR_LISTE)))
                        (COND ((IST-VARIABLE (CADR KAPAAR))
                               (SETQ KVPAAR_LISTE
                                     (CONS KAPAAR
                                           KVPAAR_LISTE)))
                              ( T  (SETQ KKPAAR_LISTE
                                         (CONS KAPAAR
                                               KKPAAR_LISTE))) )
                        (KAPAARE-ORDNEN
                             (CDR KAPAAR_LISTE)
                             KKPAAR_LISTE
                             KVPAAR_LISTE))) ))

; -KAPAARE-ORDNEN-----------------------------------------------------



  (DEFUN KNOTEN-EINBAU (KNOTEN
                        PLAN)
; ------------------------------------------------------------------
; Wert:                 Neu in den gerade bearbeiteten PLAN eingebau-
;                       ter Knoten.
;
; Globale Variablen:    --
;
; Eingangsparameter:      KNOTEN
;                       Neu erzeugter Planknoten, der in PLAN einge-
;                       baut, d.h. an die Knotenliste von PLAN hinzuge-
;                       fuegt werden soll.
;
;                         PLAN
;                       Gerade bearbeiteter Plan, in den KNOTEN einge-
;                       baut werden soll.
;
; Nebeneffekte:         Einbau von KNOTEN in PLAN, d.h. Hinzufuegen von
;                       KNOTEN an die Knotenliste von PLAN.
;
; ruft auf:             --
;
; aufgerufen von:       ERZEUGER-ERFUELLT-MERKMAL
;                       RETTER-NEUTRALISIERT-ZERSTOERER
;
; -KNOTEN-EINBAU------------------------------------------------------

        (SETF (PLAN-KNOTENLISTE PLAN)
              (CONS
                  KNOTEN
                  (PLAN-KNOTENLISTE PLAN)))
        KNOTEN)

; -KNOTEN-EINBAU------------------------------------------------------
```

```
  (DEFUN LOKALE-CONSTRAINTS-AUSWERTEN (PLANKNOTEN
                                       ARGUMENT)
; ---------------------------------------------------------------------
; Wert:                 - Falls die Operator-Instanz PLANKNOTEN lokale
;                         Constraints aufweist, die die moeglichen
;                         Werte fuer das uebergebene ARGUMENT ein-
;                         schraenken:
;                           Liste zweielementiger Unterlisten (= Kno-
;                           ten-Argument-Paare) mit folgendem Aufbau:
;                             1. Element: Knoten (= PLANKNOTEN),
;                             2. Element: Argument, das zum uebergebe-
;                                         nen ARGUMENT explizit lokal
;                                         noncodesigniert ist.
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ARGUMENT (-> PLANKNOTEN)
;                       Argument (mit Angabe des Bezug-Planknotens zur
;                       eindeutigen Kennzeichnung), dessen lokal non-
;                       codesignierte "Partner-Argumente" bestimmt
;                       werden sollen.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             LOKALE-CONSTRAINTS-AUSWERTEN-HILF
;
; aufgerufen von:       BILDE-NONCOD-KAPAARE
;
; -LOKALE-CONSTRAINTS-AUSWERTEN-----------------------------------------

        (LOKALE-CONSTRAINTS-AUSWERTEN-HILF
                   (OPERATOR-LOKALE_CONSTRAINTS
                          (EVAL (KNOTEN-OPERATOR
                                 PLANKNOTEN)))
                   PLANKNOTEN
                   ARGUMENT))

; -LOKALE-CONSTRAINTS-AUSWERTEN-----------------------------------------



  (DEFUN LOKALE-CONSTRAINTS-AUSWERTEN-HILF (CONSTRAINTLISTE
                                            PLANKNOTEN
                                            ARGUMENT)
; ---------------------------------------------------------------------
; Wert:                 - Falls die Operator-Instanz PLANKNOTEN lokale
;                         Constraints aufweist, die die moeglichen
;                         Werte fuer das uebergebene ARGUMENT ein-
;                         schraenken:
;                           Liste zweielementiger Unterlisten (= Kno-
;                           ten-Argument-Paare) mit folgendem Aufbau:
;                             1. Element: Knoten (= PLANKNOTEN),
;                             2. Element: Argument, das zum uebergebe-
;                                         nen ARGUMENT explizit lokal
;                                         noncodesigniert ist.
;                       - Sonst:
;                           NIL.
;
```

```
; Globale Variablen:    --
;
; Eingangsparameter:      CONSTRAINTLISTE
;                       Liste der noch auszuwertenden lokalen (Non-
;                       codesignation-)Constraints der Operator-
;                       Instanz PLANKNOTEN.
;
;                         ARGUMENT (-> PLANKNOTEN)
;                       Argument (mit Angabe des Bezug-Planknotens zur
;                       eindeutigen Kennzeichnung), dessen lokal non-
;                       codesignierte "Partner-Argumente" bestimmt
;                       werden sollen.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       LOKALE-CONSTRAINTS-AUSWERTEN
;
; -LOKALE-CONSTRAINTS-AUSWERTEN-HILF---------------------------------

       (COND ((NULL CONSTRAINTLISTE) NIL)
             ( T  (LET* ((CONSTRAINTARGUMENTE
                                 (CDAR CONSTRAINTLISTE))
                         (CONSTRAINTARGUMENT1
                                 (CAR CONSTRAINTARGUMENTE))
                         (CONSTRAINTARGUMENT2
                                 (CADR CONSTRAINTARGUMENTE)))
                        (APPEND
                            (COND ((EQUAL ARGUMENT
                                          CONSTRAINTARGUMENT1)
                                   (LIST
                                      (LIST PLANKNOTEN
                                            CONSTRAINTARGUMENT2)))
                                  ((EQUAL ARGUMENT
                                          CONSTRAINTARGUMENT2)
                                   (LIST
                                      (LIST PLANKNOTEN
                                            CONSTRAINTARGUMENT1))) )
                            (LOKALE-CONSTRAINTS-AUSWERTEN-HILF
                                      (CDR CONSTRAINTLISTE)
                                      PLANKNOTEN
                                      ARGUMENT))) )))

; -LOKALE-CONSTRAINTS-AUSWERTEN-HILF---------------------------------



  (DEFUN MATCH-MOEGLICH (MERKMAL1
                         MERKMAL2)
; ------------------------------------------------------------------
; Wert:                 - T, falls die beiden uebergebenen Merkmale
;                         MERKMAL1 und MERKMAL2 gemaess einfachem Mu-
;                         stervergleich codesignierbar sind.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMAL1, MERKMAL2
;                       Zu vergleichendes Merkmalpaar.
```

```
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       IST-VARIABLE
;
; aufgerufen von:       sich selbst
;                       HOLE-NEUE-ERZEUGER-HILF
;
; -MATCH-MOEGLICH-----------------------------------------------------

       (COND ((AND
                  (NULL MERKMAL1)
                  (NULL MERKMAL2))
              T)
             ((OR
                  (NULL MERKMAL1)
                  (NULL MERKMAL2))
              NIL)
             ( T  (LET ((ELEMENT1    (CAR MERKMAL1))
                        (ELEMENT2    (CAR MERKMAL2)))
                       (COND ((OR
                                 (EQUAL ELEMENT1
                                        ELEMENT2)
                                 (AND
                                    (IST-VARIABLE ELEMENT1)
                                    (NOT (NULL ELEMENT2)))
                                 (AND
                                    (IST-VARIABLE ELEMENT2)
                                    (NOT (NULL ELEMENT1))) )
                              (MATCH-MOEGLICH
                                     (CDR MERKMAL1)
                                     (CDR MERKMAL2))) ))) ))

; -MATCH-MOEGLICH-----------------------------------------------------



  (DEFUN MEHRFACHE-RAUS (LISTE)
; ---------------------------------------------------------------------
; Wert:                 Liste mit denselben Elementen wie die ueberge-
;                       bene LISTE bis auf solche Elemente, die in der
;                       uebergebenen LISTE mehrfach (gemaess EQUAL)
;                       vertreten sind; letztere tauchen in der Resul-
;                       tatliste jeweils nur einmal auf und zwar posi-
;                       tionsmaessig repraesentiert durch ihren ersten
;                       Vertreter in der uebergebenen LISTE (von
;                       rechts her gesehen).
;
; Globale Variablen:    --
;
; Eingangsparameter:      LISTE
;                       Liste, an Hand der eine Kopie erzeugt werden
;                       soll, wobei mehrfach vorhandene Elemente (ge-
;                       maess EQUAL) im Original nur einmal in der Ko-
;                       pie vertreten sein sollen.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             sich selbst
;
```

```
; aufgerufen von:       sich selbst
;                       BESTIMME-DIREKTE-ABINDUNGEN
;                       BESTIMME-DIREKTE-VORGAENGER
;                       BESTIMME-VARIABLEN
;                       BESTIMME-VORGAENGER
;
; -MEHRFACHE-RAUS-----------------------------------------------------

        (COND ((NULL LISTE) NIL)
              ( T  (LET ((ELEMENT
                                (CAR LISTE))
                         (RESTLISTE
                                (CDR LISTE)))
                        (APPEND
                            (COND ((NOT (MEMBER ELEMENT
                                                RESTLISTE
                                                :TEST 'EQUAL))
                                   (LIST ELEMENT)))
                            (MEHRFACHE-RAUS
                                 RESTLISTE))) )))

; -MEHRFACHE-RAUS-----------------------------------------------------



  (DEFUN MERKMALE-CODESIGNIERBAR (KNOTEN1
                                  MERKMAL1
                                  KNOTEN2
                                  MERKMAL2)
; ------------------------------------------------------------------
; Wert:                 - Falls die Vorbedingung MERKMAL1 der Opera-
;                         tor-Instanz KNOTEN1 mit der Vorbedingung
;                         MERKMAL2 der Operatorinstanz KNOTEN2 code-
;                         signierbar ist:
;                           Liste mit einer dreielementigen Unterliste
;                           mit folgendem Aufbau:
;                             1. Element: Kennung  (= 'ALT),
;                             2. Element: Erzeuger (= KNOTEN1),
;                             3. Element: Effekt   (= MERKMAL1).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMAL1 (-> KNOTEN1),
;                         MERKMAL2 (-> KNOTEN2)
;                       Merkmalpaar, das auf Codesignierbarkeit zu un-
;                       tersuchen ist.
;
;                       (Anm.: Zwei Merkmale sind genau dann codesig-
;                              nierbar, wenn saemtliche Argumente
;                              paarweise codesignierbar sind.)
;
; Nebeneffekte:         --
;
; ruft auf:             MERKMALE-CODESIGNIERBAR-HILF
;
; aufgerufen von:       IST-ERZEUGER-HILF
;                       IST-ZERSTOERER-HILF
;
; -MERKMALE-CODESIGNIERBAR--------------------------------------------
```

```
        (COND ((EQUAL (CAR MERKMAL1)
                      (CAR MERKMAL2))
               (MERKMALE-CODESIGNIERBAR-HILF
                      KNOTEN1
                      MERKMAL1
                      (CDR MERKMAL1)
                      KNOTEN2
                      (CDR MERKMAL2))) ))

; -MERKMALE-CODESIGNIERBAR-------------------------------------------



  (DEFUN MERKMALE-CODESIGNIERBAR-HILF (KNOTEN1
                                       MERKMAL1
                                       MERKMALARGUMENTE1
                                       KNOTEN2
                                       MERKMALARGUMENTE2)
; ------------------------------------------------------------------
; Wert:                - Falls saemtliche Argumente in MERKMALARGU-
;                        MENTE1 der Operatorinstanz KNOTEN1 mit den
;                        entsprechenden Argumenten in MERKMALARGUMEN-
;                        TE2 der Operatorinstanz KNOTEN2 codesignier-
;                        bar sind:
;                          Liste mit einer dreielementigen Unterliste
;                          mit folgendem Aufbau:
;                            1. Element: Kennung  (= 'ALT),
;                            2. Element: Erzeuger (= KNOTEN1),
;                            3. Element: Effekt   (= MERKMAL1).
;                      - Sonst:
;                          NIL.
;
; Globale Variablen:   --
;
; Eingangsparameter:     MERKMALARGUMENTE1 (-> KNOTEN1),
;                        MERKMALARGUMENTE2 (-> KNOTEN2)
;                      Argumentlisten, welche daraufhin zu untersu-
;                      chen sind, ob alle korrespondierenden Elemente
;                      codesignierbar sind.
;
;                        MERKMAL1
;                      Vorbedingung von KNOTEN1, die im Erfolgsfall
;                      als drittes Element der Resultatliste zurueck-
;                      gegeben wird.
;
; Nebeneffekte:        --
;
; ruft auf:            sich selbst
;                      IST-VARIABLE
;                      ARGUMENTE-NONCODESIGNIERT
;
; aufgerufen von:      sich selbst
;                      MERKMALE-CODESIGNIERBAR
;
; -MERKMALE-CODESIGNIERBAR-HILF--------------------------------------

      (COND ((AND
                (NULL MERKMALARGUMENTE1)
                (NULL MERKMALARGUMENTE2))
             (LIST
```

```
                (LIST 'ALT
                      KNOTEN1
                      MERKMAL1)))
            ((OR
                (NULL MERKMALARGUMENTE1)
                (NULL MERKMALARGUMENTE2))
             NIL)
            ( T  (LET ((ARGUMENT1
                          (CAR MERKMALARGUMENTE1))
                       (ARGUMENT2
                          (CAR MERKMALARGUMENTE2)))
                      (COND ((AND
                                 (NOT (EQUAL ARGUMENT1
                                             ARGUMENT2))
                                 (OR
                                    (NULL ARGUMENT1)
                                    (NULL ARGUMENT2)
                                    (AND
                                       (NOT (IST-VARIABLE
                                                   ARGUMENT1))
                                       (NOT (IST-VARIABLE
                                                   ARGUMENT2))) ))
                             NIL)
                            ((OR
                                (AND
                                   (EQUAL ARGUMENT1
                                          ARGUMENT2)
                                   (NOT (IST-VARIABLE
                                                ARGUMENT1)))
                                (NOT (ARGUMENTE-NONCODESIGNIERT
                                             KNOTEN1
                                             ARGUMENT1
                                             KNOTEN2
                                             ARGUMENT2)))
                             (MERKMALE-CODESIGNIERBAR-HILF
                                    KNOTEN1
                                    MERKMAL1
                                    (CDR MERKMALARGUMENTE1)
                                    KNOTEN2
                                    (CDR MERKMALARGUMENTE2))) ))) ))

; -MERKMALE-CODESIGNIERBAR-HILF--------------------------------------



  (DEFUN MERKMALE-NONCODESIGNIERBAR (KNOTEN1
                                     MERKMAL1
                                     KNOTEN2
                                     MERKMAL2)
; ------------------------------------------------------------------
; Wert:                 - T, falls die beiden uebergebenen Merkmale
;                         MERKMAL1 (-> KNOTEN1) und MERKMAL2 (-> KNO-
;                         TEN2) noncodesignierbar sind.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMAL1 (-> KNOTEN1), MERKMAL2 (-> KNOTEN2)
;                       Merkmalpaar (mit Angabe des jeweiligen Bezug-
;                       Planknotens zur eindeutigen Kennzeichnung),
;                       das auf Noncodesignierbarkeit untersucht wer-
;                       den soll.
```

```
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       ARGUMENTE-CODESIGNIERT
;
; aufgerufen von:       sich selbst
;                       KONFLIKT-LOESUNGEN
;
; -MERKMALE-NONCODESIGNIERBAR----------------------------------------

       (COND ((OR
                 (NULL MERKMAL1)
                 (NULL MERKMAL2))
              NIL)
             ((OR
                 (NOT (ARGUMENTE-CODESIGNIERT
                             KNOTEN1
                             (CAR MERKMAL1)
                             KNOTEN2
                             (CAR MERKMAL2)))
                 (MERKMALE-NONCODESIGNIERBAR
                        KNOTEN1
                        (CDR MERKMAL1)
                        KNOTEN2
                        (CDR MERKMAL2))) )))

; -MERKMALE-NONCODESIGNIERBAR----------------------------------------




  (DEFUN NUTZER-ABHAENGIGKEIT-EINTRAGEN (PLAN
                                         NUTZER
                                         MERKMAL
                                         ERZEUGER)
; ------------------------------------------------------------------
; Wert:                 Plan (mit demselben Speicherbereich wie PLAN),
;                       der sich von PLAN nur darin unterscheidet,
;                       dass die Anhaengigkeiten von NUTZER bezueglich
;                       ERZEUGER auf den aktuellen Stand gebracht
;                       sind.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAN
;                       Modifizierter, aber noch nicht vollstaendig
;                       aktualisierter Plan.
;
;                         NUTZER
;                       Aktuell betrachteter ungeloester Knoten.
;
;                         MERKMAL
;                       Aktuell betrachtete unerfuellte Vorbedingung
;                       von NUTZER.
;
;                         ERZEUGER
;                       Fuer die unerfuellte Vorbedingung MERKMAL des
;                       Planknotens NUTZERS gewaehlter erzeugender
;                       Planknoten.
;
```

```
; Nebeneffekte:         --
;
; ruft auf:             SUCHE-ZERSTOERER
;                       ERZEUGER-ETABLIEREN
;                       ERZEUGER-VORMERKEN
;
; aufgerufen von:       ORDNUNG-VERSCHAERFEN
;                       RETTER-NEUTRALISIERT-ZERSTOERER
;                       SEPARIEREN
;
; -NUTZER-ABHAENGIGKEIT-EINTRAGEN-----------------------------------

        (COND ((NULL (SUCHE-ZERSTOERER
                              (PLAN-KNOTENLISTE
                                      PLAN)
                              NUTZER
                              MERKMAL
                              ERZEUGER))
               (ERZEUGER-ETABLIEREN
                      NUTZER
                      MERKMAL
                      ERZEUGER))
              ( T  (ERZEUGER-VORMERKEN
                         NUTZER
                         MERKMAL
                         ERZEUGER)))
        (LIST PLAN))

; -NUTZER-ABHAENGIGKEIT-EINTRAGEN-----------------------------------



  (DEFUN OBJEKT-BESTIMMUNG (OBJEKTNAME
                            MODUS
                            OBJEKTLISTE)
; ------------------------------------------------------------------
; Wert:                 - Erster Knoten bzw. erstes Wertconstraint mit
;                         dem Namen OBJEKTNAME innerhalb der Knoten-
;                         bzw. Wertconstraintliste OBJEKTLISTE, falls
;                         darin existent.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      OBJEKTNAME
;                       Name des Knotens bzw. Wertconstraints, der
;                       bzw. das in der Knoten- bzw. Wertconstraintli-
;                       ste OBJEKTLISTE bestimmt werden soll.
;
;                         MODUS
;                       Indikator fuer den Objekttyp (= 'KNOTEN-NAME
;                       oder 'WERTCONSTRAINT-NAME).
;
;                         OBJEKTLISTE
;                       Liste von Knoten, bzw. Wertconstraints, inner-
;                       halb der nach einem Knoten bzw. Wertconstraint
;                       mit dem Namen OBJEKTNAME ermittelt werden
;                       soll.
;
; Nebeneffekte:         --
;
```

```
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       KNOTEN-KOPIEREN
;                       VERWEISE-KOPIEREN
;                       VERWEISE-MIT-MERKMAL-KOPIEREN
;
; -OBJEKT-BESTIMMUNG-------------------------------------------------

        (COND ((NULL OBJEKTLISTE) NIL)
              ( T  (LET ((OBJEKT   (CAR OBJEKTLISTE)))
                        (COND ((EQUAL OBJEKTNAME
                                      (FUNCALL MODUS OBJEKT))
                               OBJEKT)
                              ( T  (OBJEKT-BESTIMMUNG
                                          OBJEKTNAME
                                          MODUS
                                          (CDR OBJEKTLISTE))) ))) ))

; -OBJEKT-BESTIMMUNG-------------------------------------------------



  (DEFUN SUCHE-ALTE-ERZEUGER (PLANKNOTENLISTE
                              NUTZER
                              MERKMAL)
; ------------------------------------------------------------------
; Wert:                 - Falls Erzeuger im aktuell betrachteten Plan
;                         mit der Knotenliste PLANKNOTENLISTE exi-
;                         stent:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung (= 'ALT),
;                             2. Element: Erzeuger(-Knoten), welcher
;                                         fuer den Nutzer(-Knoten) die
;                                         noch unerfuellte Vorbedin-
;                                         gung MERKMAL erzeugen koenn-
;                                         te,
;                             3. Element: Effekt (Nachbedingung des
;                                         Erzeugers, welche mit der
;                                         Vorbedingung MERKMAL des
;                                         NUTZERs codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTENLISTE
;                       Knotenliste des aktuell betrachteten Planes.
;
;                         NUTZER
;                       Ungeloester Knoten innerhalb des aktuell be-
;                       trachteten Plans, fuer dessen noch unerfuell-
;                       te Vorbedingung MERKMAL nach Erzeugern ge-
;                       sucht wird.
;
;                         MERKMAL
;                       Unerfuelltes (Vorbedingungs-)Merkmal von NUT-
;                       ZER, fuer dessen Erfuellung nach Erzeugern ge-
;                       sucht wird.
;
```

```
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             BESTIMME-ZIELKNOTEN
;                       BESTIMME-DIREKTE-VORGAENGER
;                       SUCHE-ALTE-ERZEUGER-HILF
;
; aufgerufen von:       ERMITTLE-ERZEUGER
;
; -SUCHE-ALTE-ERZEUGER----------------------------------------------

       (LET* ((ZIELKNOTEN
                    (BESTIMME-ZIELKNOTEN
                                PLANKNOTENLISTE))
              (GENERATION
                    (BESTIMME-DIREKTE-VORGAENGER
                                ZIELKNOTEN))
              (NAECHSTE_GENERATION
                    (BESTIMME-DIREKTE-VORGAENGER
                                GENERATION)))
             (SUCHE-ALTE-ERZEUGER-HILF
                      GENERATION
                      NAECHSTE_GENERATION
                      NUTZER
                      MERKMAL)))

; -SUCHE-ALTE-ERZEUGER----------------------------------------------



  (DEFUN SUCHE-ALTE-ERZEUGER-HILF (GENERATION
                                   NAECHSTE_GENERATION
                                   NUTZER
                                   MERKMAL
                                   &OPTIONAL K_E_M_TRIPELLISTE
                                             MARKIERTE_KNOTEN)
; -----------------------------------------------------------------
; Wert:                 - Falls Erzeuger im aktuell betrachteten Plan
;                         existent:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung (= 'ALT),
;                             2. Element: Erzeuger(-Knoten), welcher
;                                         fuer den Nutzer(-Knoten) die
;                                         noch unerfuellte Vorbedin-
;                                         gung MERKMAL erzeugen koenn-
;                                         te,
;                             3. Element: Effekt (Nachbedingung des
;                                         Erzeugers, welche mit der
;                                         Vorbedingung MERKMAL des
;                                         NUTZERs codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      GENERATION
;                       Aktuell betrachtete Knotengeneration bzw. Rest
;                       davon.
;
;                       (Anm.: Die Knoten einer Generation zeichnen
;                              sich dadurch aus, dass sie alle densel-
```

```
;                              ben Abstand zum Zielknoten des aktuell
;                              betrachteten Plans haben.)
;
;                        NAECHSTE_GENERATION
;                      Knotengeneration direkt vor GENERATION (in
;                      Richtung Startknoten).
;
;                        NUTZER
;                      Ungeloester Knoten innerhalb des aktuell be-
;                      trachteten Plans, fuer dessen noch unerfuellte
;                      Vorbedingung MERKMAL nach Erzeugern gesucht
;                      wird.
;
;                        MERKMAL
;                      Unerfuelltes (Vorbedingungs-)Merkmal von NUT-
;                      ZER, fuer dessen Erfuellung nach Erzeugern ge-
;                      sucht wird.
;
;                        K_E_M_TRIPELLISTE (optional!)
;                      Liste der gefundenen bereits im aktuell be-
;                      trachteten Plan eingebauten Erzeuger, wobei
;                      diese Liste obig erwaehnten Aufbau der Resul-
;                      tatliste aufweist.
;
;                        MARKIERTE_KNOTEN
;                      Liste der bereits untersuchten Knoten.
;
;                      K_E_M_TRIPELLISTE sowie MARKIERTE_KNOTEN sind
;                      als optionale Parameter definiert, um eine ex-
;                      plizite Initialisierung mit NIL beim erstmali-
;                      gen Aufruf von aussen mit derselben Wirkung zu
;                      umgehen.
;
; Nebeneffekte:        Erzeugen der Resultatliste.
;
; ruft auf:            sich selbst
;                      BESTIMME-DIREKTE-VORGAENGER
;                      IST-VORGAENGER
;                      IST-ERZEUGER
;
; aufgerufen von:      sich selbst
;                      SUCHE-ALTE-ERZEUGER
;
; -SUCHE-ALTE-ERZEUGER-HILF-----------------------------------------

       (COND ((AND
                  (NULL GENERATION)
                  (NULL NAECHSTE_GENERATION))
              K_E_M_TRIPELLISTE)
             ((NULL GENERATION)
              (LET ((KANDIDATEN
                          (REMOVE-IF
                             #'(LAMBDA (PLANKNOTEN)
                                   (MEMBER PLANKNOTEN
                                           MARKIERTE_KNOTEN))
                             NAECHSTE_GENERATION)))
                (SUCHE-ALTE-ERZEUGER-HILF
                          KANDIDATEN
                          (BESTIMME-DIREKTE-VORGAENGER
                                    KANDIDATEN)
                          NUTZER
                          MERKMAL
```

```
                              K_E_M_TRIPELLISTE
                              (APPEND
                                  KANDIDATEN
                                  MARKIERTE_KNOTEN))) )
             ( T  (LET ((BETRACHTETER_KNOTEN
                               (CAR GENERATION)))
                        (SUCHE-ALTE-ERZEUGER-HILF
                             (CDR GENERATION)
                             NAECHSTE_GENERATION
                             NUTZER
                             MERKMAL
                             (APPEND
                                 (AND
                                    (NOT (EQL BETRACHTETER_KNOTEN
                                              NUTZER))
                                    (NOT (IST-VORGAENGER
                                              NUTZER
                                              BETRACHTETER_KNOTEN))
                                    (IST-ERZEUGER
                                           BETRACHTETER_KNOTEN
                                           NUTZER
                                           MERKMAL))
                                K_E_M_TRIPELLISTE)
                             MARKIERTE_KNOTEN))) ))

; -SUCHE-ALTE-ERZEUGER-HILF-----------------------------------------



  (DEFUN SUCHE-UNGELOESTE-KNOTEN (PLANKNOTENLISTE)
; ---------------------------------------------------------------------
; Wert:                 Liste saemtlicher ungeloester Planknoten der
;                       dem Zielknoten des aktuell betrachteten Planes
;                       am naechsten gelegenen Knoten-Generation mit
;                       mindestens einem ungeloesten Planknoten.
;
;                       (Anm.: SUCHE-UNGELOESTE-KNOTEN wird nur dann
;                              aufgerufen, wenn feststeht, dass im ak-
;                              tuell betrachteten Plan noch mindestens
;                              ein ungeloester Knoten vorhanden ist!)
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTENLISTE
;                       Knotenliste des aktuell betrachteten Planes.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             BESTIMME-ZIELKNOTEN
;                       GELOEST
;                       BESTIMME-DIREKTE-VORGAENGER
;                       SUCHE-UNGELOESTE-KNOTEN-HILF
;
; aufgerufen von:       PLAN-ERWEITERUNG
;
; -SUCHE-UNGELOESTE-KNOTEN-----------------------------------------

        (LET ((ZIELKNOTEN
                    (BESTIMME-ZIELKNOTEN
                           PLANKNOTENLISTE)))
            (COND ((NOT (GELOEST ZIELKNOTEN))
```

```
                  (LIST ZIELKNOTEN))
                 ( T  (SUCHE-UNGELOESTE-KNOTEN-HILF
                            (BESTIMME-DIREKTE-VORGAENGER
                                   ZIELKNOTEN))) )))

; -SUCHE-UNGELOESTE-KNOTEN-----------------------------------------



  (DEFUN SUCHE-UNGELOESTE-KNOTEN-HILF (GENERATION
                                       &OPTIONAL UNGELOESTE_KNOTEN
                                                 GELOESTE_KNOTEN)
; ---------------------------------------------------------------
; Wert:                 Liste saemtlicher ungeloester Planknoten der
;                       dem Zielknoten des aktuell betrachteten Planes
;                       am naechsten gelegenen Knoten-Generation mit
;                       mindestens einem ungeloesten Planknoten.
;
;                       (Anm.: SUCHE-UNGELOESTE-KNOTEN-HILF wird nur
;                              dann aufgerufen, wenn feststeht, dass
;                              im aktuell betrachteten Plan noch min-
;                              destens ein ungeloester Knoten vorhan-
;                              den ist!)
;
; Globale Variablen:    --
;
; Eingangsparameter:      GENERATION
;                       Betrachtete Knotengeneration, deren Planknoten
;                       allesamt den gleichen Abstand zum Zielknoten
;                       des aktuell betrachteten Plans aufweisen.
;
;                         UNGELOESTE_KNOTEN (optional!)
;                       Resultatliste. Liste der in GENERATION aufge-
;                       spuerten ungeloesten Planknoten.
;
;                         GELOESTE_KNOTEN (optional!)
;                       Liste der in GENERATION aufgespuerten geloe-
;                       sten Planknoten.
;                       UNGELOESTE_KNOTEN und GELOESTE_KNOTEN sind als
;                       optionale Parameter definiert, um Initialisie-
;                       rung mit NIL beim erstmaligen Aufruf von aus-
;                       sen einzusparen.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             sich selbst
;                       BESTIMME-DIREKTE-VORGAENGER
;                       GELOEST
;
; aufgerufen von:       sich selbst
;                       SUCHE-UNGELOESTE-KNOTEN
;
; -SUCHE-UNGELOESTE-KNOTEN-HILF------------------------------------

        (COND ((AND
                   (NULL GENERATION)
                   (NULL UNGELOESTE_KNOTEN))
               (SUCHE-UNGELOESTE-KNOTEN-HILF
                          (BESTIMME-DIREKTE-VORGAENGER
                                 GELOESTE_KNOTEN)))
              ((NULL GENERATION)
```

```
               UNGELOESTE_KNOTEN)
              ( T  (LET ((BETRACHTETER_KNOTEN   (CAR GENERATION)))
                        (COND ((GELOEST BETRACHTETER_KNOTEN)
                               (SETQ GELOESTE_KNOTEN
                                     (CONS BETRACHTETER_KNOTEN
                                           GELOESTE_KNOTEN)))
                              ( T  (SETQ UNGELOESTE_KNOTEN
                                         (CONS BETRACHTETER_KNOTEN
                                               UNGELOESTE_KNOTEN))) )
                        (SUCHE-UNGELOESTE-KNOTEN-HILF
                                   (CDR GENERATION)
                                   UNGELOESTE_KNOTEN
                                   GELOESTE_KNOTEN))) ))

; -SUCHE-UNGELOESTE-KNOTEN-HILF--------------------------------------




  (DEFUN WAEHLE-PLAN (PLAENE
                      &OPTIONAL GEWAEHLTER_PLAN)
; ---------------------------------------------------------------------
; Wert:                 Derjenige (unvollstaendige) nichtlineare Plan
;                       aus der Plan-Warteschlange PLAENE mit minima-
;                       ler Knoten-Anzahl. Weisen mehrere Plaene die-
;                       selbe minimale Knoten-Anzahl auf, so wird der
;                       erste Plan (vom linken Anfang der Plaene-Liste
;                       her gesehen) mit minimaler Knoten-Anzahl als
;                       Wert zurueckgeliefert.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAENE
;                       Liste mit vom Planer generierten (unvollstaen-
;                       digen) nichtlinearen Plaenen, welche eine War-
;                       teschlange der noch zu untersuchenden Plaene
;                       darstellt.
;
;                         GEWAEHLTER_PLAN (optional!)
;                       Resultatwert. Als optionaler Parameter defini-
;                       niert, um Initialisierung mit NIL bei erstma-
;                       ligem Aufruf von aussen einsparen zu koennen.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       PLANEN
;
; -WAEHLE-PLAN-------------------------------------------------------

        (COND ((NULL PLAENE)
               GEWAEHLTER_PLAN)
              ( T  (LET ((BETRACHTETER_PLAN
                                 (CAR PLAENE)))
                        (COND ((OR
                                 (NULL GEWAEHLTER_PLAN)
                                 (< (LENGTH (PLAN-KNOTENLISTE
                                                 BETRACHTETER_PLAN))
                                    (LENGTH (PLAN-KNOTENLISTE
                                                 GEWAEHLTER_PLAN))) )
```

```
                                (SETQ GEWAEHLTER_PLAN
                                      BETRACHTETER_PLAN)))
                           (WAEHLE-PLAN
                                 (CDR PLAENE)
                                 GEWAEHLTER_PLAN))) ))

; -WAEHLE-PLAN-----------------------------------------------------------




  (DEFUN WAEHLE-UNERFUELLTES-MERKMAL (MERKMALLISTE
                                      UNGELOESTER_KNOTEN
                                      &OPTIONAL
                                          GEWAEHLTES_MERKMAL
                                          ZAHL_UNGEBUNDENER_VARIABLEN)
; ---------------------------------------------------------------------
; Wert:                 Dasjenige Merkmal aus der MERKMALLISTE, das
;                       die wenigsten ungebundenen Variablen enthaelt.
;                       Liegen mehrere Merkmale mit derselben minima-
;                       len Anzahl ungebundener Variablen vor, so wird
;                       das erste Merkmal von links in MERKMALLISTE
;                       gewaehlt.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMALLISTE
;                       Liste saemtlicher unerfuellter (Vorbedin-
;                       gungs-)Merkmale von UNGELOESTER_KNOTEN.
;
;                       (Anm.: Ein Merkmal gilt dann als unerfuellt,
;                              wenn bezueglich diesem noch keine etab-
;                              lierte Abhaengigkeit eingetragen ist.)
;
;                         UNGELOESTER_KNOTEN
;                       Aktuell betrachteter ungeloester Knoten inner-
;                       halb des aktuell betrachteten Plans, dessen
;                       unerfuellte Vorbedingung (bzw. eine davon (ge-
;                       maess obiger Kriterien)) bestimmt werden soll.
;
;                         GEWAEHLTES_MERKMAL
;                       Resultatwert. Ist als optionaler Parameter zur
;                       Einsparung der Initialisierung mit NIL beim
;                       erstmaligen Aufruf von aussen definiert.
;
;                         ZAHL_UNGEBUNDENER_VARIABLEN
;                       Zahl der ungebundenen Variablen (= Konstanten
;                       oder Variablen, die mit Konstanten instanti-
;                       iert sind) des gerade gewaehlten Merkmals
;                       GEWAEHLTES_MERKMAL. Ist als optionaler Para-
;                       meter definiert, um beim erstmaligen Aufruf
;                       von aussen nicht angegeben werden zu muessen.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       ANZAHL-ARGUMENTE
;                       BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE
;
; aufgerufen von:       sich selbst
;                       PLAN-ERWEITERUNG
;
```

```
; -WAEHLE-UNERFUELLTES-MERKMAL--------------------------------------

        (COND ((NULL MERKMALLISTE)
               GEWAEHLTES_MERKMAL)
              ( T  (LET* ((BETRACHTETES_MERKMAL
                           (CAR MERKMALLISTE))
                          (ZAHL_UNGEBUNDENER_VARIABLEN_NEU
                           (- (ANZAHL-ARGUMENTE
                                    BETRACHTETES_MERKMAL)
                              (BESTIMME-ZAHL-INSTANTIIERTER-ARGUMENTE
                                        BETRACHTETES_MERKMAL
                                        UNGELOESTER_KNOTEN))) )
                          (COND
                             ((OR
                                 (NULL GEWAEHLTES_MERKMAL)
                                 (< ZAHL_UNGEBUNDENER_VARIABLEN_NEU
                                    ZAHL_UNGEBUNDENER_VARIABLEN))
                              (SETQ GEWAEHLTES_MERKMAL
                                    BETRACHTETES_MERKMAL
                                    ZAHL_UNGEBUNDENER_VARIABLEN
                                    ZAHL_UNGEBUNDENER_VARIABLEN_NEU)))
                          (WAEHLE-UNERFUELLTES-MERKMAL
                                (CDR MERKMALLISTE)
                                UNGELOESTER_KNOTEN
                                GEWAEHLTES_MERKMAL
                                ZAHL_UNGEBUNDENER_VARIABLEN))) ))

; -WAEHLE-UNERFUELLTES-MERKMAL--------------------------------------



  (DEFUN WAEHLE-UNGELOESTEN-KNOTEN (KNOTENLISTE)
; ---------------------------------------------------------------------
; Wert:                 Derjenige ungeloeste Knoten aus KNOTENLISTE,
;                       dessen Einbau in den aktuell betrachteten Plan
;                       am laengsten zurueckliegt.
;
; Globale Variablen:    --
;
; Eingangsparameter:      KNOTENLISTE
;                       Liste saemtlicher ungeloester Knoten der dem
;                       Zielknoten des aktuell betrachteten Plans am
;                       naechsten liegenden Knotengeneration mit noch
;                       ungeloesten Knoten.
;
;                       (Anm.: Saemtliche Knoten einer Knotengenera-
;                              tion weisen immer denselben Abstand zum
;                              Zielknoten auf.)
;
; Nebeneffekte:         --
;
; ruft auf:             WAEHLE-UNGELOESTEN-KNOTEN-HILF
;
; aufgerufen von:       PLAN-ERWEITERUNG
;
; -WAEHLE-UNGELOESTEN-KNOTEN----------------------------------------

        (WAEHLE-UNGELOESTEN-KNOTEN-HILF
                  (CDR KNOTENLISTE)
                  (CAR KNOTENLISTE)))
```

```
; -WAEHLE-UNGELOESTEN-KNOTEN----------------------------------------



  (DEFUN WAEHLE-UNGELOESTEN-KNOTEN-HILF (RESTKNOTENLISTE
                                         WAHL)
; ------------------------------------------------------------------
; Wert:                 Derjenige ungeloeste Knoten aus (cons WAHL
;                       RESTKNOTENLISTE), dessen Einbau in den aktuell
;                       betrachteten Plan am laengsten zurueckliegt.
;
; Globale Variablen:    --
;
; Eingangsparameter:      RESTKNOTENLISTE
;                       Noch auf ihr Alter zu betrachtende ungeloeste
;                       Knoten.
;
;                         WAHL
;                       Resultatwert. Zum gerade betrachteten Zeit-
;                       punkt derjenige Knoten, dessen Einbau unter
;                       den bislang betrachteten am laengsten zurueck-
;                       liegt.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       ALTER
;
; aufgerufen von:       sich selbst
;                       WAEHLE-UNGELOESTEN-KNOTEN
;
; -WAEHLE-UNGELOESTEN-KNOTEN-HILF-----------------------------------

        (COND ((NULL RESTKNOTENLISTE) WAHL)
              ( T  (LET ((BETRACHTETER_KNOTEN
                                (CAR RESTKNOTENLISTE)))
                         (COND ((< (ALTER BETRACHTETER_KNOTEN)
                                   (ALTER WAHL))
                                (WAEHLE-UNGELOESTEN-KNOTEN-HILF
                                         (CDR RESTKNOTENLISTE)
                                         BETRACHTETER_KNOTEN))
                               ( T  (WAEHLE-UNGELOESTEN-KNOTEN-HILF
                                            (CDR RESTKNOTENLISTE)
                                            WAHL))) ))) )

; -WAEHLE-UNGELOESTEN-KNOTEN-HILF-----------------------------------



  (DEFUN ZEICHEN-ZU-ZAHL (ZEICHEN)
; ------------------------------------------------------------------
; Wert:                 Zahl aus [0 .. 9], welche dem character
;                       ZEICHEN entspricht.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ZEICHEN
;                       Character aus [#\0 .. #\9], der in die ent-
;                       sprechende Zahl aus [0 .. 9] umzuwandeln ist.
;
```

```
; Nebeneffekte:         --
;
; ruft auf:             --
;
; aufgerufen von:       ALTER
;
; -ZEICHEN-ZU-ZAHL-----------------------------------------------------

        (COND ((EQUAL ZEICHEN '#\1) 1)
              ((EQUAL ZEICHEN '#\2) 2)
              ((EQUAL ZEICHEN '#\3) 3)
              ((EQUAL ZEICHEN '#\4) 4)
              ((EQUAL ZEICHEN '#\5) 5)
              ((EQUAL ZEICHEN '#\6) 6)
              ((EQUAL ZEICHEN '#\7) 7)
              ((EQUAL ZEICHEN '#\8) 8)
              ((EQUAL ZEICHEN '#\9) 9)
              ( T  0)))

; -ZEICHEN-ZU-ZAHL-----------------------------------------------------



  (DEFUN ZEITCONSTRAINTNETZ-OPTIMIEREN (PLANKNOTENLISTE
                                        KNOTEN_VON
                                        KNOTEN_NACH)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTENLISTE
;                       Knotenliste des gerade bearbeiteten Planes.
;
;                         KNOTEN_VON
;                       Knoten, von dem der neu gesetzte Vorgaenger-
;                       Verweis ausgeht.
;
;                         KNOTEN_NACH
;                       Knoten, auf den der neu gesetzte Vorgaenger-
;                       Verweis zeigt.
;
; Nebeneffekte:         Optimieren des Zeit-Constraintnetzes des gera-
;                       de bearbeiteten Plans insofern, dass redun-
;                       dante Vorgaenger-Verweise geloescht werden.
;
; ruft auf:             BESTIMME-ZIELKNOTEN
;                       BESTIMME-DIREKTE-VORGAENGER
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF
;
; aufgerufen von:       EINTRAEGE-AKTUALISIEREN-E
;                       EINTRAEGE-AKTUALISIEREN-R
;                       ORDNUNG-VERSCHAERFEN
;
; -ZEITCONSTRAINTNETZ-OPTIMIEREN---------------------------------------

        (LET ((ZIELKNOTEN
                    (BESTIMME-ZIELKNOTEN
                              PLANKNOTENLISTE)))
             (COND ((NOT (EQL KNOTEN_VON
                              ZIELKNOTEN))
```

```
                    (ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF
                            (LIST ZIELKNOTEN)
                            (BESTIMME-DIREKTE-VORGAENGER
                                   ZIELKNOTEN)
                            KNOTEN_VON
                            KNOTEN_NACH)))
              (SETF (KNOTEN-VORGAENGER KNOTEN_VON)
                    (REMOVE-IF
                        #'(LAMBDA (PLANKNOTEN)
                               (IST-VORGAENGER
                                      PLANKNOTEN
                                      KNOTEN_NACH))
                        (KNOTEN-VORGAENGER KNOTEN_VON))) ))

; -ZEITCONSTRAINTNETZ-OPTIMIEREN-------------------------------------



  (DEFUN ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF (GENERATION
                                             NAECHSTE_GENERATION
                                             KNOTEN_VON
                                             KNOTEN_NACH)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      GENERATION
;                       Jeweils betrachtete Knotengeneration (bzw.
;                       Rest davon) des gerade bearbeiteten Planes.
;
;                         NAECHSTE_GENERATION
;                       Direkte Vorgaengergeneration von GENERATION.
;
;                         KNOTEN_VON
;                       Knoten, von dem der neu gesetzte Vorgaenger-
;                       Verweis ausgeht.
;
;                         KNOTEN_NACH
;                       Knoten, auf den der neu gesetzte Vorgaenger-
;                       Verweis zeigt.
;
; Nebeneffekte:         Optimieren des Zeit-Constraintnetzes des gera-
;                             de bearbeiteten Plans insofern, dass redun-
;                       dante Vorgaenger-Verweise (mit Ausnahme der
;                       Vorgaenger-Eintraege von KNOTEN_VON) geloescht
;                       werden.
;
; ruft auf:             sich selbst
;                       BESTIMME-DIREKTE-VORGAENGER
;                       IST-VORGAENGER
;
; aufgerufen von:       sich selbst
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN
;
; -ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF--------------------------------

        (COND ((AND
                   (NULL GENERATION)
                   (NULL NAECHSTE_GENERATION)))
              ((NULL GENERATION)
```

```
                (ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF
                       NAECHSTE_GENERATION
                       (BESTIMME-DIREKTE-VORGAENGER
                              NAECHSTE_GENERATION)
                       KNOTEN_VON
                       KNOTEN_NACH))
               ( T  (LET ((PLANKNOTEN
                                (CAR GENERATION)))
                         (COND ((IST-VORGAENGER
                                       KNOTEN_VON
                                       PLANKNOTEN)
                                (SETF (KNOTEN-VORGAENGER
                                                PLANKNOTEN)
                                      (REMOVE-IF
                                           #'(LAMBDA (KNOTEN)
                                                 (OR
                                                   (EQL KNOTEN
                                                        KNOTEN_NACH)
                                                   (IST-VORGAENGER
                                                        KNOTEN
                                                        KNOTEN_NACH)))
                                           (KNOTEN-VORGAENGER
                                                     PLANKNOTEN))) ))
                               (ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF
                                      (CDR GENERATION)
                                      NAECHSTE_GENERATION
                                      KNOTEN_VON
                                      KNOTEN_NACH))) ))

; -ZEITCONSTRAINTNETZ-OPTIMIEREN-HILF--------------------------------


; =ALLGEMEINE PLANUNGSFUNKTIONEN=====================================
```

```
; ======================================================================
; ============      FUNKTIONEN ZUR KONFLIKT-BEHEBUNG      ============
; ======================================================================

; Stand: 29.05.1990


  (DEFUN EINTRAEGE-AKTUALISIEREN-R (PLAN
                                    NUTZER
                                    RETTER
                                    RETTER_EFFEKT
                                    ZERSTOERER
                                    ZERSTOERER_EFFEKT)
; ----------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAN
;                       Gerade bearbeiteter Plan, innerhalb dem ein
;                       Konflikt entdeckt worden ist, der mit Hilfe
;                       des RETTER-Knotens (White Knight) behoben
;                       wird.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des gerade in
;                       Bearbeitung befindlichen PLANs.
;
;                         RETTER
;                       Gewaehlter Knoten (alt oder neu) des gerade
;                       bearbeiteten PLANs , der in der Lage ist, die
;                       negative Wirkung des entdeckten ZERSTOERERs
;                       dadurch aufzuheben, dass er auch die gewaehlte
;                       unerfuellte Vorbedingung des NUTZERs erzeugt.
;                       Die Nachbedingung von RETTER, die diesen Ef-
;                       fekt bewirkt, wird nun aber nicht per Codesig-
;                       nation-Wertconstraint an das unerfuellte Vor-
;                       bedingungs-Merkmal von NUTZER, sondern an den
;                       nichtnegierten Part des den Konflikt verursa-
;                       chenden Nachbedingungs-Merkmals von ZERSTOERER
;                       gebunden!
;
;                         RETTER_EFFEKT
;                       Nichtnegiertes Nachbedingungs-Merkmal von RET-
;                       TER, vom dem aus die Konflikt loesende Wirkung
;                       ausgeht dadurch, dass es mit der gewaehlten
;                       unerfuellten Vorbedingung von NUTZER codesig-
;                       nierbar ist.
;
;                         ZERSTOERER
;                       Knoten des gerade bearbeiteten PLANs, der die
;                       erzeugende Wirkung des bereits bestimmten Er-
;                       zeugers fuer NUTZER (moeglicherweise) aufheben
;                       koennte und somit einen Konflikt hervorruft.
;
;                         ZERSTOERER_EFFEKT
;                       Negiertes Nachbedingungs-Merkmal von ZERSTOE-
;                       RER, von dem aus die Konflikt verursachende
;                       Wirkung ausgeht dadurch, dass sein nichtne-
;                       gierter Part mit der gewaehlten unerfuellten
;                       Vorbedingung von NUTZER codesignierbar ist.
```

```
;
; Nebeneffekte:         - Falls RETTER noch kein Vorgaenger von NUTZER
;                         ist:
;                         Eintragen von RETTER als direkter Vorgaenger
;                         von NUTZER in die Vorgaengerliste von NUTZER
;                         mit anschliessender Zeitconstraintnetz-Opti-
;                         mierung.
;                       - Falls ZERSTOERER noch kein Vorgaenger von
;                         RETTER ist:
;                         Eintragen von ZERSTOERER als direkter Vor-
;                         gaenger von RETTER in die Vorgaengerliste
;                         von RETTER mit anschliessender Zeitcon-
;                         straintnetz-Optimierung.
;                       - Erzeugen eines neuen Wertconstraints mit
;                         Einbau in den gerade bearbeiteten PLAN, um
;                         RETTER_EFFEKT (-> RETTER) und nichtnegierter
;                         Part von ZERSTOERER_EFFEKT (-> ZERSTOERER)
;                         explizit zu codesignieren.
;                       - Eintragen von RETTER als Erzeuger fuer NUT-
;                         ZER in das KNOTEN-erzeugt_fuer-Feld von RET-
;                         TER, wenn auch RETTER nur als "White Knight"
;                         (zur Zerstoerer-Neutralisierung), und nicht
;                         als etablierter Erzeuger fungiert!
;
; ruft auf:             IST-VORGAENGER
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN
;                       CONSTRAINT-EINBAU
;
; aufgerufen von:       RETTER-NEUTRALISIERT-ZERSTOERER
;
; -EINTRAEGE-AKTUALISIEREN-R-----------------------------------------

       (LET ((WERTCONSTRAINT
                  NIL)
             (KNOTENLISTE
                  (PLAN-KNOTENLISTE PLAN)))
            (COND ((NOT (IST-VORGAENGER
                              RETTER
                              NUTZER))
                   (SETF (KNOTEN-VORGAENGER NUTZER)
                         (CONS
                             RETTER
                             (KNOTEN-VORGAENGER NUTZER)))
                   (ZEITCONSTRAINTNETZ-OPTIMIEREN
                           KNOTENLISTE
                           NUTZER
                           RETTER)))
            (COND ((NOT (IST-VORGAENGER
                              ZERSTOERER
                              RETTER))
                   (SETF (KNOTEN-VORGAENGER RETTER)
                         (CONS
                             ZERSTOERER
                             (KNOTEN-VORGAENGER RETTER)))
                   (ZEITCONSTRAINTNETZ-OPTIMIEREN
                           KNOTENLISTE
                           RETTER
                           ZERSTOERER)))
            (SETQ WERTCONSTRAINT_NEU
                  (CONSTRAINT-EINBAU
                        RETTER
                        RETTER_EFFEKT
```

```
                          ZERSTOERER
                          ZERSTOERER_EFFEKT
                          PLAN))
             (SETF (KNOTEN-CODESIGNATIONS ZERSTOERER)
                   (CONS
                       WERTCONSTRAINT_NEU
                       (KNOTEN-CODESIGNATIONS ZERSTOERER)))
             (SETF (KNOTEN-CODESIGNATIONS RETTER)
                   (CONS
                       WERTCONSTRAINT_NEU
                       (KNOTEN-CODESIGNATIONS RETTER)))
             (SETF (KNOTEN-erzeugt_fuer RETTER)
                   (CONS
                       (LIST NUTZER
                             RETTER_EFFEKT)
                       (KNOTEN-erzeugt_fuer RETTER))) ))

; -EINTRAEGE-AKTUALISIEREN-R---------------------------------------



  (DEFUN ERMITTLE-RETTER (PLANKNOTENLISTE
                          NUTZER
                          MERKMAL
                          ERZEUGER
                          ZERSTOERER)
; ---------------------------------------------------------------------
; Wert:                 Liste dreielementiger Unterlisten mit folgen-
;                       dem Aufbau:
;                         1. Element: Kennung
;                                     (= 'ALT, falls Retter bereits im
;                                              Plan eingebaut ist;
;                                     (= 'NEU, falls Retter noch neu,
;                                              also noch nicht im Plan
;                                              eingebaut ist),
;                         2. Element: Retter(-Knoten), welcher fuer
;                                     NUTZER die noch unerfuellte Vor-
;                                     bedingung MERKMAL erzeugen
;                                     koennte und daher die negative
;                                     Wirkung des ZERSTOERERs aufheben
;                                     kann,
;                         3. Element: Effekt
;                                     (Nichtnegierte Nachbedingung des
;                                      RETTERs, welche mit der uner-
;                                      fuellten Vorbedingung  MERKMAL
;                                      (-> NUTZER) und dem nichtnegier-
;                                      ten Part der Konflikt verursa-
;                                      chenden Nachbedingung des ZER-
;                                      STOERERs codesignierbar ist).
;
; Globale Variablen:    VERFUEGBARE_OPERATOREN
;
; Eingangsparameter:      PLANKNOTENLISTE
;                 Knotenliste der gerade bearbeiteten Plans.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des gerade be-
;                       arbeiteten Plans.
;
;                         MERKMAL
;                       Gewaehlte unerfuellte Vorbedingung von NUTZER.
```

```
;
;                         ERZEUGER
;                       Knoten des gerade bearbeiteten Plans, der als
;                       Erzeuger fuer MERKMAL (-> NUTZER) vorgemerkt,
;                       aber Konflikt bedingt noch nicht etabliert
;                       ist.
;
;                         ZERSTOERER
;                       Knoten des gerade bearbeiteten Plans, der die
;                       Abhaengigkeit ERZEUGER-MERKMAL-NUTZER stoert.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             SUCHE-ALTE-RETTER
;                       IST-ZERSTOERER
;                       HOLE-NEUE-ERZEUGER
;
; aufgerufen von:       RETTER-EINSCHIEBEN
;
; -ERMITTLE-RETTER-----------------------------------------------------

        (APPEND
            (SUCHE-ALTE-RETTER
                     PLANKNOTENLISTE
                     NUTZER
                     MERKMAL
                     ERZEUGER
                     ZERSTOERER)
            (REMOVE-IF
                 #'(LAMBDA (K_R_M_TRIPEL)
                         (IST-ZERSTOERER
                                 (CADR K_R_M_TRIPEL)
                                 NUTZER
                                 MERKMAL))
                (HOLE-NEUE-ERZEUGER
                         VERFUEGBARE_OPERATOREN
                         MERKMAL))) )

; -ERMITTLE-RETTER-----------------------------------------------------



  (DEFUN ERZEUGER-VORGEMERKT (ABHAENGIGKEITEN
                              MERKMAL)
; ---------------------------------------------------------------------
; Wert:                 - Falls Erzeuger vorgemerkt, d.h. als Erzeuger
;                         in der Abhaengigkeitsliste des Nutzers ein-
;                         getragen, aber Konflikt bedingt noch nicht
;                         etabliert ist:
;                           Liste, deren einziges Element folgende Ab-
;                           haengigkeit darstellt:
;                         ('NOCH_NICHT_ETABLIERT <Erzeuger> <Merkmal>)
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ABHAENGIGKEITEN
;                       Liste saemtlicher Abhaengigkeiten des Nutzers
;                       (= (KNOTEN-ist_abhaengig_von Nutzer) ).
;
```

```
;                        MERKMAL
;                      Gewaehlte unerfuellte Vorbedingung des Nut-
;                      zers.
;
; Nebeneffekte:        Erzeugen der Resultatliste.
;
; ruft auf:            sich selbst
;
; aufgerufen von:      sich selbst
;                      ERMITTLE-ERZEUGER
;                      ERZEUGER-ETABLIEREN
;                      ERZEUGER-VORMERKEN
;
; -ERZEUGER-VORGEMERKT-------------------------------------------------

       (COND ((NULL ABHAENGIGKEITEN) NIL)
             ( T  (LET ((ABHAENGIGKEIT
                              (CAR ABHAENGIGKEITEN)))
                       (COND ((AND
                                 (= (LENGTH ABHAENGIGKEIT)
                                    3)
                                 (EQUAL (CADDR ABHAENGIGKEIT)
                                        MERKMAL))
                              (LIST ABHAENGIGKEIT))
                             ( T  (ERZEUGER-VORGEMERKT
                                      (CDR ABHAENGIGKEITEN)
                                      MERKMAL))) ))) )

; -ERZEUGER-VORGEMERKT-------------------------------------------------



  (DEFUN ERZEUGER-VORMERKEN (NUTZER
                             MERKMAL
                             ERZEUGER)
; ---------------------------------------------------------------------
; Wert:                irrelevant!
;
; Globale Variablen:   --
;
; Eingangsparameter:     NUTZER
;                      Gewaehlter ungeloester Knoten des gerade bear-
;                      beiteten Plans.
;
;                        MERKMAL
;                      Gewaehlte unerfuellte Vorbedingung von NUTZER.
;
;                        ERZEUGER
;                      Als Erzeuger fuer MERKMAL (-> NUTZER) bestimm-
;                      ter Knoten des gerade bearbeiteten Plans, der
;                      in der Abhaengigkeitsliste von NUTZER Konflikt
;                      bedingt nicht etabliert, sondern vorerst nur
;                      vorgemerkt werden soll.
;
; Nebeneffekte:        - Falls ERZEUGER nicht bereits vorgemerkt ist:
;                        Eintragen von ERZEUGER als vorgemerkter Er-
;                        zeuger in die Abhaengigkeitsliste von NUT-
;                        ZER. Der Eintrag, der zu den bereits beste-
;                        henden Abhaengigkeiten von NUTZER hinzuge-
;                        fuegt wird, weist folgende Gestalt auf:
;                         `(NOCH_NICHT_ETABLIERT ,ERZEUGER ,MERKMAL).
```

```
;
; ruft auf:             ERZEUGER-VORGEMERKT
;
; aufgerufen von:       NUTZER-ABHAENGIGKEIT-EINTRAGEN
;                       ABHAENGIGKEITEN-PRUEFEN
;                       NUTZER-ABHAENGIGKEIT-EINTRAGEN
;
; -ERZEUGER-VORMERKEN-------------------------------------------------

        (LET ((ABHAENGIGKEITEN
                    (KNOTEN-ist_abhaengig_von NUTZER)))
             (COND ((NOT (ERZEUGER-VORGEMERKT
                              ABHAENGIGKEITEN
                              MERKMAL))
                    (SETF (KNOTEN-ist_abhaengig_von NUTZER)
                          (CONS
                              (LIST 'NOCH_NICHT_ETABLIERT
                                    ERZEUGER
                                    MERKMAL)
                              ABHAENGIGKEITEN))) )))

; -ERZEUGER-VORMERKEN-------------------------------------------------



  (DEFUN IST-ZERSTOERER (MOEGLICHER_ZERSTOERER
                         NUTZER
                         MERKMAL)
; ---------------------------------------------------------------------
; Wert:                 - Falls MOEGLICHER_ZERSTOERER ein potentieller
;                         Zerstoerer bezueglich der Erzeugung von
;                         MERKMAL fuer NUTZER darstellt:
;                           Zweielementige Liste mit folgendem Aufbau:
;                             1. Element: Zerstoerer-Knoten
;                                         (= MOEGLICHER_ZERSTOERER),
;                             2. Element: Zerstoerer-Effekt
;                                         (= Nichtnegierter Part des
;                                            negierten Nachbedingungs-
;                                            Merkmals von MOEGLICHER_
;                                            ZERSTOERER, der mit der
;                                            unerfuellten Vorbedingung
;                                            MERKMAL von NUTZER code-
;                                            signierbar ist).
;                       - Sonst:
;                           NIL.
;
;                       (Anm.:  IST-ZERSTOERER betrachtet MOEGLICHER_
;                               ZERSTOERER isoliert und nicht im Plan-
;                               knoten-Kontext. Wird ein Wert <> NIL
;                               zurueckgeliefert, so ist dies eine
;                               notwendige, aber keine hinreichende
;                               Bedingung fuer einen Zerstoerer (moeg-
;                               licher- oder notwendigerweise). Ein
;                               Konflikt liegt erst dann vor, wenn
;                               IST-ZERSTOERER fuer einen betrachteten
;                               Knoten einen Wert <> NIL liefert und
;                               die bestehende (partielle) Operator-
;                               Ordnung im betrachteten Plan eine Kon-
;                               flikt-Situation zulaesst!)
;
; Globale Variablen:    --
```

```
;
; Eingangsparameter:      MOEGLICHER_ZERSTOERER
;                       Knoten des gerade bearbeiteten Plans, der da-
;                       raufhin zu untersuchen ist, ob er ein poten-
;                       tieller Zerstoerer bezueglich der Erzeugung
;                       von MERKMAL fuer NUTZER darstellt. Dies ist
;                       dann der Fall, wenn MOEGLICHER_ZERSTOERER eine
;                       negierte Nachbedingung besitzt, deren nichtne-
;                       gierter Part mit MERKMAL codesignierbar ist.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des gerade bear-
;                       beiteten Plans.
;
;                         MERKMAL
;                       Gewaehlte noch unerfuellte Vorbedingung von
;                       NUTZER.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             IST-ZERSTOERER-HILF
;                       BESTIMME-NEGIERTE-EFFEKTE
;
; aufgerufen von:       ERMITTLE-RETTER
;                       SUCHE-ZERSTOERER-HILF
;
; -IST-ZERSTOERER-----------------------------------------------------

       (IST-ZERSTOERER-HILF
               MOEGLICHER_ZERSTOERER
               (BESTIMME-NEGIERTE-EFFEKTE
                     (KNOTEN-OPERATOR
                              MOEGLICHER_ZERSTOERER))
               NUTZER
               MERKMAL))

; -IST-ZERSTOERER-----------------------------------------------------



  (DEFUN IST-ZERSTOERER-HILF (MOEGLICHER_ZERSTOERER
                              ZERSTOERER_EFFEKTE
                              NUTZER
                              MERKMAL)
; ------------------------------------------------------------------
; Wert:                 - Falls MOEGLICHER_ZERSTOERER ein potentieller
;                         Zerstoerer bezueglich der Erzeugung von
;                         MERKMAL fuer NUTZER darstellt:
;                           Zweielementige Liste mit folgendem Aufbau:
;                             1. Element: Zerstoerer-Knoten
;                                         (= MOEGLICHER_ZERSTOERER),
;                             2. Element: Zerstoerer-Effekt
;                                         (= Nichtnegierter Part des
;                                            negierten Nachbedingungs-
;                                            Merkmals von MOEGLICHER_
;                                            ZERSTOERER, der mit der
;                                            unerfuellten Vorbedingung
;                                            MERKMAL von NUTZER code-
;                                            signierbar ist; sollte
;                                            dies fuer mehrere Nachbe-
;                                            dingungen aus ZERSTOERER_
```

```
;                                          EFFEKTE gelten, so wird
;                                          die erste von links ge-
;                                          waehlt).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MOEGLICHER_ZERSTOERER
;                       Knoten des gerade bearbeiteten Plans, der da-
;                       raufhin zu untersuchen ist, ob er ein poten-
;                       tieller Zerstoerer bezueglich der Erzeugung
;                       von MERKMAL fuer NUTZER darstellt. Dies ist
;                       dann der Fall, wenn MOEGLICHER_ZERSTOERER eine
;                       negierte Nachbedingung besitzt, deren nichtne-
;                       gierter Part mit MERKMAL codesignierbar ist.
;
;                         ZERSTOERER_EFFEKTE
;                       Liste der nichtnegierten Parts (saemtlicher)
;                       negierter Nachbedingungs-Merkmale von MOEG-
;                       LICHER_ZERSTOERER.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des gerade bear-
;                       beiteten Plans.
;
;                         MERKMAL
;                       Gewaehlte noch unerfuellte Vorbedingung von
;                       NUTZER.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             sich selbst
;                       MERKMALE-CODESIGNIERBAR
;                       RETTER-NACHGESCHALTET
;
; aufgerufen von:       sich selbst
;                       IST-ZERSTOERER
;                       ABHAENGIGKEITEN-PRUEFEN
;
; -IST-ZERSTOERER-HILF-----------------------------------------------

       (COND ((NULL ZERSTOERER_EFFEKTE) NIL)
             ( T  (LET* ((ZERSTOERER_EFFEKT
                               (CAR ZERSTOERER_EFFEKTE))
                          (ZERSTOERER_TEST
                               (MERKMALE-CODESIGNIERBAR
                                       MOEGLICHER_ZERSTOERER
                                       ZERSTOERER_EFFEKT
                                       NUTZER
                                       MERKMAL)))
                        (COND ((AND
                                   ZERSTOERER_TEST
                                   (NOT (RETTER-NACHGESCHALTET
                                             MOEGLICHER_ZERSTOERER
                                             ZERSTOERER_EFFEKT
                                             NUTZER)))
                               (CDAR ZERSTOERER_TEST))
                              ( T  (IST-ZERSTOERER-HILF
                                        MOEGLICHER_ZERSTOERER
                                        (CDR ZERSTOERER_EFFEKTE)
                                        NUTZER
```

```
                                        MERKMAL))) ))) )

; -IST-ZERSTOERER-HILF-----------------------------------------------



  (DEFUN KONFLIKT-ANALYSIEREN (ZERSTOERER
                               NUTZER
                               ERZEUGER)
; ---------------------------------------------------------------------
; Wert:                 - 'PARALLEL-KONFLIKT, falls ein Parallel-Kon-
;                         flikt vorliegt. Dies ist dann der Fall, wenn
;                         ERZEUGER kein Vorgaenger-Knoten von ZERSTOE-
;                         RER und ZERSTOERER kein Vorgaenger-Knoten
;                         von NUTZER ist.
;                             - 'GABEL-KONFLIKT_1, falls ein Gabel-Konflikt
;                         Typ 1 vorliegt. Dies ist dann der Fall, wenn
;                         ZERSTOERER Vorgaenger-Knoten von NUTZER,
;                         aber ERZEUGER kein Vorgaenger-Knoten von
;                         ZERSTOERER ist.
;                       - 'GABEL-KONFLIKT_2, falls ein Gabel-Konflikt
;                         Typ 2 vorliegt. Dies ist dann der Fall, wenn
;                         ERZEUGER Vorgaenger-Knoten von ZERSTOERER,
;                         aber ZERSTOERER kein Vorgaenger-Knoten von
;                         NUTZER ist.
;                       - 'LINEAR-KONFLIKT, falls ein Linear-Konflikt
;                         vorliegt. Dies ist dann der Fall, wenn ER-
;                         ZEUGER Vorgaenger-Knoten von ZERSTOERER und
;                         ZERSTOERER Vorgaenger-Knoten von NUTZER ist.
;
;                       (Anm.:  Wenn KONFLIKT-ANALYSIEREN aufgerufen
;                               wird, liegt auf jeden Fall einer die-
;                               ser Konflikt-Typen vor. Falls z.B.
;                               ZERSTOERER ein Vorgaenger-Knoten von
;                               ERZEUGER oder NUTZER ein Vorgaenger-
;                               Knoten von ZERSTOERER waere, so laege
;                               ueberhaupt keine Konflikt-Situation
;                               vor und KONFLIKT-ANALYSIEREN wuerde
;                               nicht aufgerufen werden!)
;
; Globale Variablen:    --
;
; Eingangsparameter:      ZERSTOERER
;                       Knoten des gerade bearbeiteten Plans, der die
;                       Abhaengigkeit ERZEUGER <-> NUTZER stoert.
;
;                         NUTZER
;                       Gewaehlter noch ungeloester Knoten des gerade
;                       bearbeiteten Plans.
;
;                         ERZEUGER
;                       Zur Erzeugung der gewaehlten unerfuellten Vor-
;                       bedingung von NUTZER bestimmter Knoten des ge-
;                       rade bearbeiteten Plans.
;
; Nebeneffekte:         --
;
; ruft auf:             IST-VORGAENGER
;
; aufgerufen von:       KONFLIKT-LOESUNGEN
;
```

```
; -KONFLIKT-ANALYSIEREN-----------------------------------------------

        (LET ((ERZEUGER_VOR_ZERSTOERER
                        (IST-VORGAENGER
                                ERZEUGER
                                ZERSTOERER))
              (ZERSTOERER_VOR_NUTZER
                        (IST-VORGAENGER
                                ZERSTOERER
                                NUTZER)))
             (COND ((AND
                        (NOT ERZEUGER_VOR_ZERSTOERER)
                        (NOT ZERSTOERER_VOR_NUTZER))
                    'PARALLEL-KONFLIKT)
                   ((AND
                        (NOT ERZEUGER_VOR_ZERSTOERER)
                        ZERSTOERER_VOR_NUTZER)
                    'GABEL-KONFLIKT_1)
                   ((AND
                        ERZEUGER_VOR_ZERSTOERER
                        (NOT ZERSTOERER_VOR_NUTZER))
                    'GABEL-KONFLIKT_2)
                   ( T  'LINEAR-KONFLIKT))) )

; -KONFLIKT-ANALYSIEREN-----------------------------------------------



  (DEFUN ORDNUNG-VERSCHAERFEN (MODUS
                               PLAN_ALT
                               ERZEUGER_ALT
                               ZERSTOERER_ALT
                               NUTZER_ALT
                               MERKMAL)
; ---------------------------------------------------------------------
; Wert:                 Liste mit dem folgenden einen Element:
;                         Aktualisierter Plan (mit eigenem internen
;                         Speicherbereich), der sich von PLAN_ALT da-
;                         rin unterscheidet, dass der zuletzt erkannte
;                         Konflikt dadurch behoben worden ist, dass
;                         der Zerstoerer-Knoten vor den Erzeuger (Vor-
;                         verlegung; Demotion) bzw. hinter den NUTZER
;                         (Nachverlegung; Promotion) - je nach MODUS -
;                         verlegt wurde.
;
;                       (Anm.:  Ein Plan sei hier als aktualisiert be-
;                               zeichnet, wenn er in sich konsistent
;                               ist und saemtliche Eintraege, die
;                               Plan-Verwaltung betreffend, vollzogen
;                               sind!)
;
; Globale Variablen:    --
;
; Eingangsparameter:      MODUS
;                       Indikator dafuer, ob Demotion (MODUS = 'VOR-
;                       VERLEGEN) oder Promotion (MODUS = 'NACHVERLE-
;                       GEN) als Konflikt-Loesungsmethode gewaehlt
;                       werden soll.
;
;                         PLAN_ALT
;                       Gerade bearbeiteter Plan, in dem ein Konflikt
```

```
;                       erkannt wurde.
;
;                         ERZEUGER_ALT
;                       Zur Erfuellung der Vorbedingung MERKMAL von
;                       NUTZER bestimmter Knoten aus PLAN_ALT.
;
;                         ZERSTOERER_ALT
;                       Knoten aus PLAN_ALT, der die Abhaengigkeit
;                       ERZEUGER_ALT <-> NUTZER_ALT stoert.
;
;                         NUTZER_ALT
;                       Gewaehlter noch ungeloester Knoten aus PLAN_
;                       ALT.
;
;                         MERKMAL
;                       Gewaehlte noch unerfuellte Vorbedingung von
;                       NUTZER_ALT.
;
; Nebeneffekte:         Erzeugen einer neuen Plan-Struktur (= einzi-
;                       ge Komponente der Resultatliste) und der Re-
;                       sultatliste.
;
; ruft auf:             KNOTEN-KOPIEREN
;                       PLAN-KOPIEREN
;                       ZEITCONSTRAINTNETZ-OPTIMIEREN
;                       NUTZER-ABHAENGIGKEIT-EINTRAGEN
;
; aufgerufen von:       KONFLIKT-LOESUNGEN
;
; -ORDNUNG-VERSCHAERFEN-----------------------------------------------

        (LET* ((PLAN_NEU
                       (PLAN-KOPIEREN
                                PLAN_ALT))
               (KNOTENLISTE_NEU
                       (PLAN-KNOTENLISTE
                                PLAN_NEU))
               (ERZEUGER_NEU
                       (KNOTEN-KOPIEREN
                                ERZEUGER_ALT
                                KNOTENLISTE_NEU))
               (ZERSTOERER_NEU
                       (KNOTEN-KOPIEREN
                                ZERSTOERER_ALT
                                KNOTENLISTE_NEU))
               (NUTZER_NEU
                       (KNOTEN-KOPIEREN
                                NUTZER_ALT
                                KNOTENLISTE_NEU))
               (KNOTEN_VON
                       ERZEUGER_NEU)
               (KNOTEN_NACH
                       ZERSTOERER_NEU))
              (COND ((EQUAL MODUS
                            'NACHVERLEGEN)
                     (SETQ KNOTEN_VON  ZERSTOERER_NEU
                           KNOTEN_NACH NUTZER_NEU)))
              (SETF (KNOTEN-VORGAENGER KNOTEN_VON)
                    (CONS
                        KNOTEN_NACH
                        (KNOTEN-VORGAENGER KNOTEN_VON)))
              (ZEITCONSTRAINTNETZ-OPTIMIEREN
```

```
                      KNOTENLISTE_NEU
                      KNOTEN_VON
                      KNOTEN_NACH)
               (NUTZER-ABHAENGIGKEIT-EINTRAGEN
                      PLAN_NEU
                      NUTZER_NEU
                      MERKMAL
                      ERZEUGER_NEU)))

; -ORDNUNG-VERSCHAERFEN-----------------------------------------------



  (DEFUN RETTER-EINSCHIEBEN (PLAN_ALT
                             ERZEUGER_ALT
                             ZERSTOERER_ALT
                             ZERSTOERER_EFFEKT
                             NUTZER_ALT
                             MERKMAL)
; ---------------------------------------------------------------------
; Wert:                 Liste aktualisierter Plaene (jeweils mit eige-
;                       nem internen Speicherbereich), die sich je-
;                       weils von PLAN_ALT darin unterscheiden, dass
;                       der zuletzt erkannte Konflikt dadurch behoben
;                       worden ist, dass der Zerstoerer-Knoten jeweils
;                       durch verschiedene "Retter"-Knoten (alt oder
;                       neu) neutralisiert worden ist.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAN_ALT
;                       Gerade bearbeiteter Plan, in dem ein Konflikt
;                       erkannt wurde.
;
;                         ERZEUGER_ALT
;                       Zur Erfuellung der Vorbedingung MERKMAL (->
;                       NUTZER_ALT) bestimmter Knoten aus PLAN_ALT.
;
;                         ZERSTOERER_ALT
;                       Knoten aus PLAN_ALT, der die Abhaengigkeit
;                       ERZEUGER_ALT <-> NUTZER_ALT stoert.
;
;                         ZERSTOERER_EFFEKT
;                       Nichtnegierter Part der negierten Nachbedin-
;                       gung von ZERSTOERER_ALT, der mit der Vorbe-
;                       dingung MERKMAL (-> NUTZER_ALT) codesignierbar
;                       ist.
;
;                         NUTZER_ALT
;                       Gewaehlter noch ungeloester Knoten aus PLAN_
;                       ALT.
;
;                         MERKMAL
;                       Gewaehlte noch unerfuellte Vorbedingung von
;                       NUTZER_ALT.
;
; Nebeneffekte:         - Erzeugen der Retter-Liste (= Kennung-Retter-
;                         Rettereffekt-Tripelliste).
;                       - Erzeugen einer Menge neuer aktualisierter
;                         Plaene (mit jeweils eigenem Speicherbe-
;                         reich), die durch Einschalten eines (jeweils
```

```
;                         verschiedenen) Retter-Knotens um einen Kon-
;                         flikt aermer sind als PLAN_ALT, und Verpak-
;                         ken derselben in eine Liste, die Resultatli-
;                         ste.
;
; ruft auf:             RETTER-NEUTRALISIERT-ZERSTOERER
;                       ERMITTLE-RETTER
;
; aufgerufen von:       KONFLIKT-LOESUNGEN
;
; -RETTER-EINSCHIEBEN-----------------------------------------------

       (RETTER-NEUTRALISIERT-ZERSTOERER
                  PLAN_ALT
                  ERZEUGER_ALT
                  ZERSTOERER_ALT
                  ZERSTOERER_EFFEKT
                  NUTZER_ALT
                  MERKMAL
                  (ERMITTLE-RETTER
                        (PLAN-KNOTENLISTE
                                 PLAN_ALT)
                        NUTZER_ALT
                        MERKMAL
                        ERZEUGER_ALT
                        ZERSTOERER_ALT)))

; -RETTER-EINSCHIEBEN-----------------------------------------------



  (DEFUN RETTER-NACHGESCHALTET (ZERSTOERER
                                ZERSTOERER_EFFEKT
                                NUTZER)
; ------------------------------------------------------------------
; Wert:                 - T, falls ZERSTOERER durch einen nachgeschal-
;                         teten Retter-Knoten bereits neutralisiert
;                         ist.
;                       - NIL, sonst.
;
;                       (Anm.:  Bei Aufruf von RETTER-NACHGESCHALTET
;                               steht noch nicht fest, ob eine Kon-
;                               flikt-Situation vorliegt. Wird als Re-
;                               sultatwert NIL geliefert, dann steht
;                               fest, dass ZERSTOERER einen Konflikt
;                               hervorruft. Der Resultatwert NIL ist
;                               eine notwendige Bedingung fuer einen
;                               Konflikt.)
;
; Globale Variablen:    --
;
; Eingangsparameter:      ZERSTOERER
;                       Knoten des gerade bearbeiteten Plans, der,
;                       falls kein Retter nachgeschaltet ist, was RET-
;                       TER-NACHGESCHALTET ueberprueft, die Abhaengig-
;                       keit Erzeuger <-> NUTZER stoert.
;
;                         ZERSTOERER_EFFEKT
;                       Nichtnegierter Part der negierten Nachbedin-
;                       gung von ZERSTOERER, der mit der betrachteten
;                       unerfuellten Vorbedingung von NUTZER codesig-
```

```
;                       nierbar ist.
;
;                         NUTZER
;                       Gewaehlter noch ungeloester Knoten des gerade
;                       bearbeiteten Plans.
;
; Nebeneffekte:         Bestimmen saemtlicher Vorgaenger-Knoten von
;                       NUTZER.
;
; ruft auf:             BESTIMME-VORGAENGER
;                       RETTER-NACHGESCHALTET-HILF
;
; aufgerufen von:       IST-ZERSTOERER-HILF
;
; -RETTER-NACHGESCHALTET-----------------------------------------------

       (RETTER-NACHGESCHALTET-HILF
                  ZERSTOERER
                  (KNOTEN-CODESIGNATIONS
                            ZERSTOERER)
                  ZERSTOERER_EFFEKT
                  (BESTIMME-VORGAENGER
                            NUTZER)
                  NUTZER))

; -RETTER-NACHGESCHALTET-----------------------------------------------



  (DEFUN RETTER-NACHGESCHALTET-HILF (ZERSTOERER
                                     ZERSTOERER_CODESIGNATIONS
                                     ZERSTOERER_EFFEKT
                                     NUTZER_VORGAENGER
                                     NUTZER)
; ---------------------------------------------------------------------
; Wert:                 - T, falls ZERSTOERER durch einen nachgeschal-
;                         teten Retter-Knoten bereits neutralisiert
;                         ist.
;                       - NIL, sonst.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ZERSTOERER
;                       Knoten des gerade bearbeiteten Plans, der,
;                       falls kein Retter nachgeschaltet ist, die Ab-
;                       haengigkeit Erzeuger <-> NUTZER stoert.
;
;                         ZERSTOERER_CODESIGNATIONS
;                       Liste (saemtlicher) Wertconstraints, welche
;                       im CODESIGNATIONS-Feld von ZERSTOERER stehen.
;
;                         ZERSTOERER_EFFEKT
;                       Nichtnegierter Part der negierten Nachbedin-
;                       gung von ZERSTOERER, der mit der betrachteten
;                       unerfuellten Vorbedingung von NUTZER codesig-
;                       nierbar ist.
;
;                         NUTZER_VORGAENGER
;                       Liste saemtlicher Vorgaenger-Knoten von NUTZER
;                       im gerade bearbeiteten Plan.
;
```

```
;                        NUTZER
;                      Gewaehlter noch ungeloester Knoten des gerade
;                      bearbeiteten Plans.
;
; Nebeneffekte:        --
;
; ruft auf:            sich selbst
;                      IST-VORGAENGER
;
; aufgerufen von:      sich selbst
;                      RETTER-NACHGESCHALTET
;
; -RETTER-NACHGESCHALTET-HILF---------------------------------------

       (COND ((NULL ZERSTOERER_CODESIGNATIONS) NIL)
             ( T  (LET* ((CODESIGNATION
                              (CAR ZERSTOERER_CODESIGNATIONS))
                         (COD_KNOTEN
                              (WERTCONSTRAINT-KNOTEN
                                        CODESIGNATION))
                         (COD_MERKMALE
                              (WERTCONSTRAINT-MERKMALPAAR
                                        CODESIGNATION))
                         (EFFEKT
                              (CADR COD_MERKMALE))
                         (MOEGLICHER_RETTER
                              (COND ((EQL ZERSTOERER
                                          (CAR COD_KNOTEN))
                                     (CADR COD_KNOTEN))
                                    ( T  (SETQ EFFEKT
                                               (CAR COD_MERKMALE))
                                         (CAR COD_KNOTEN))) ))
                        (OR
                           (AND
                              (MEMBER ZERSTOERER_EFFEKT
                                      COD_MERKMALE)
                              (MEMBER EFFEKT
                                      (BESTIMME-NICHTNEGIERTE-EFFEKTE
                                            (KNOTEN-OPERATOR
                                                 MOEGLICHER_RETTER)))
                              (IST-VORGAENGER
                                      MOEGLICHER_RETTER
                                      NUTZER
                                      NUTZER_VORGAENGER)
                              (IST-VORGAENGER
                                      ZERSTOERER
                                      MOEGLICHER_RETTER))
                           (RETTER-NACHGESCHALTET-HILF
                                    ZERSTOERER
                                    (CDR ZERSTOERER_CODESIGNATIONS)
                                    ZERSTOERER_EFFEKT
                                    NUTZER_VORGAENGER
                                    NUTZER))) )))

; -RETTER-NACHGESCHALTET-HILF---------------------------------------



  (DEFUN RETTER-NEUTRALISIERT-ZERSTOERER (PLAN_ALT
                                          ERZEUGER_ALT
                                          ZERSTOERER_ALT
```

```
                                        ZERSTOERER_EFFEKT
                                        NUTZER_ALT
                                        MERKMAL
                                        K_R_M_TRIPELLISTE
                                        &OPTIONAL PLAENE)
; ---------------------------------------------------------------------
; Wert:                 Liste aktualisierter Plaene (jeweils mit eige-
;                       nem internen Speicherbereich), die sich je-
;                       weils von PLAN_ALT darin unterscheiden, dass
;                       der zuletzt erkannte Konflikt dadurch behoben
;                       worden ist, dass der Zerstoerer-Knoten jeweils
;                       durch verschiedene "Retter"-Knoten (alt oder
;                       neu) neutralisiert worden ist.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAN_ALT
;                       Gerade bearbeiteter Plan, in dem ein Konflikt
;                       erkannt wurde.
;
;                         ERZEUGER_ALT
;                       Zur Erfuellung der Vorbedingung MERKMAL (->
;                       NUTZER_ALT) bestimmter Knoten aus PLAN_ALT.
;
;                         ZERSTOERER_ALT
;                       Knoten aus PLAN_ALT, der die Abhaengigkeit
;                       ERZEUGER_ALT <-> NUTZER_ALT stoert.
;
;                         ZERSTOERER_EFFEKT
;                       Nichtnegierter Part der negierten Nachbedin-
;                       gung von ZERSTOERER_ALT, der mit der Vorbe-
;                       dingung MERKMAL (-> NUTZER_ALT) codesignierbar
;                       ist.
;
;                         NUTZER_ALT
;                       Gewaehlter noch ungeloester Knoten aus PLAN_
;                       ALT.
;
;                         MERKMAL
;                       Gewaehlte noch unerfuellte Vorbedingung von
;                       NUTZER_ALT.
;
;                         K_R_M_TRIPELLISTE
;                       Liste dreielementiger Unterlisten (=Kennung-
;                       Retter-Rettereffekt-Tripel) mit folgendem
;                       Aufbau:
;                         1. Element: Kennung
;                                     (= 'ALT oder 'NEU),
;                         2. Element: Retter-Knoten
;                                     (zur Neutralisierung von ZER-
;                                      STOERER_ALT),
;                         3. Element: Retter-Effekt
;                                     (= Nichtnegierte Nachbedingung
;                                        des Retters, die sowohl mit
;                                        MERKMAL (-> NUTZER_ALT), als
;                                        auch mit ZERSTOERER_EFFEKT (->
;                                        ZERSTOERER_ALT) codesignierbar
;                                        ist).
;
;                         PLAENE (optional!)
;                       Resultatliste. Ist als optionaler Parameter
;                       definiert, um Initialisierung mit NIL beim
```

```
;                       erstmaligen Aufruf von aussen einzusparen.
;
;
; Nebeneffekte:         Erzeugen einer Menge neuer aktualisierter
;                       Plaene (mit jeweils eigenem Speicherbereich),
;                       die durch Einschalten eines (jeweils verschie-
;                       denen) Retter-Knotens um einen Konflikt aermer
;                       sind als PLAN_ALT, und Verpacken derselben in
;                       eine Liste, die Resultatliste.
;
; ruft auf:             sich selbst
;                       PLAN-KOPIEREN
;                       KNOTEN-EINBAU
;                       KNOTEN-KOPIEREN
;                       EINTRAEGE-AKTUALISIEREN-R
;                       ABHAENGIGKEITEN-AKTUALISIEREN
;                       NUTZER-ABHAENGIGKEIT-EINTRAGEN
;
; aufgerufen von:       sich selbst
;                       RETTER-EINSCHIEBEN
;
; -RETTER-NEUTRALISIERT-ZERSTOERER-----------------------------------

       (COND ((NULL K_R_M_TRIPELLISTE)
              (NREVERSE PLAENE))
             ( T  (LET* ((K_R_M_TRIPEL
                              (CAR K_R_M_TRIPELLISTE))
                         (KENNUNG
                              (CAR K_R_M_TRIPEL))
                         (RETTER_ALT
                              (CADR K_R_M_TRIPEL))
                         (RETTER_EFFEKT
                              (CADDR K_R_M_TRIPEL))
                         (PLAN_NEU
                              (PLAN-KOPIEREN
                                       PLAN_ALT))
                         (NEU
                              (EQUAL KENNUNG
                                     'NEU))
                         (RETTER_NEU
                              (COND (NEU
                                     (KNOTEN-EINBAU
                                              RETTER_ALT
                                              PLAN_NEU))
                                    ( T
                                     (KNOTEN-KOPIEREN
                                              RETTER_ALT
                                              (PLAN-KNOTENLISTE
                                                  PLAN_NEU))) ))
                         (KNOTENLISTE_NEU
                              (PLAN-KNOTENLISTE
                                       PLAN_NEU))
                         (NUTZER_NEU
                              (KNOTEN-KOPIEREN
                                       NUTZER_ALT
                                       KNOTENLISTE_NEU))
                         (ERZEUGER_NEU
                              (KNOTEN-KOPIEREN
                                       ERZEUGER_ALT
                                       KNOTENLISTE_NEU))
                         (ZERSTOERER_NEU
                              (KNOTEN-KOPIEREN
```

```
                                        ZERSTOERER_ALT
                                        KNOTENLISTE_NEU)))
                          (EINTRAEGE-AKTUALISIEREN-R
                                  PLAN_NEU
                                  NUTZER_NEU
                                  RETTER_NEU
                                  RETTER_EFFEKT
                                  ZERSTOERER_NEU
                                  ZERSTOERER_EFFEKT)
                          (COND (NEU
                                 (ABHAENGIGKEITEN-AKTUALISIEREN
                                          RETTER_NEU
                                          KNOTENLISTE_NEU)))
                          (NUTZER-ABHAENGIGKEIT-EINTRAGEN
                                  PLAN_NEU
                                  NUTZER_NEU
                                  MERKMAL
                                  ERZEUGER_NEU)
                          (RETTER-NEUTRALISIERT-ZERSTOERER
                                  PLAN_ALT
                                  ERZEUGER_ALT
                                  ZERSTOERER_ALT
                                  ZERSTOERER_EFFEKT
                                  NUTZER_ALT
                                  MERKMAL
                                  (CDR K_R_M_TRIPELLISTE)
                                  (CONS
                                      PLAN_NEU
                                      PLAENE))) )))

; -RETTER-NEUTRALISIERT-ZERSTOERER-----------------------------------



  (DEFUN SEPARIEREN (PLAN_ALT
                     ERZEUGER_ALT
                     ZERSTOERER_ALT
                     ZERSTOERER_EFFEKT
                     NUTZER_ALT
                     MERKMAL)
; ------------------------------------------------------------------
; Wert:                 Liste mit dem einen folgenden Element:
;                         Aktualisierter Plan (mit eigenem internen
;                         Speicherbereich), der sich von PLAN_ALT da-
;                         rin unterscheidet, dass der zuletzt erkannte
;                         Konflikt dadurch behoben worden ist, dass
;                         zwischen Zerstoerer-Effekt und Nutzer-Merk-
;                         mal ein Noncodesignation-Wertconstraint ge-
;                         setzt wurde.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAN_ALT
;                       Gerade bearbeiteter Plan, in dem ein Konflikt
;                       erkannt wurde.
;
;                         ERZEUGER_ALT
;                       Zur Erfuellung der Vorbedingung MERKMAL (->
;                       NUTZER_ALT) bestimmter Knoten aus PLAN_ALT.
;
;                         ZERSTOERER_ALT
```

```
;                       Knoten aus PLAN_ALT, der die Abhaengigkeit
;                       ERZEUGER_ALT <-> NUTZER_ALT stoert.
;
;                         ZERSTOERER_EFFEKT
;                       Nichtnegierter Part der negierten Nachbedin-
;                       gung von ZERSTOERER_ALT, der mit der Vorbe-
;                       dingung MERKMAL (-> NUTZER_ALT) codesignierbar
;                       ist.
;
;                         NUTZER_ALT
;                       Gewaehlter noch ungeloester Knoten aus PLAN_
;                       ALT.
;
;                         MERKMAL
;                       Gewaehlte noch unerfuellte Vorbedingung von
;                       NUTZER_ALT.
;
; Nebeneffekte:         Erzeugen einer neuen Plan-Struktur (= einzi-
;                       ge Komponente der Resultatliste) und der Re-
;                       sultatliste.
;
; ruft auf:             PLAN-KOPIEREN
;                       KNOTEN-KOPIEREN
;                       CONSTRAINT-EINBAU
;                       NUTZER-ABHAENGIGKEIT-EINTRAGEN
;
; aufgerufen von:       KONFLIKT-LOESUNGEN
;
; -SEPARIEREN-----------------------------------------------------------

       (LET* ((PLAN_NEU
                      (PLAN-KOPIEREN
                               PLAN_ALT))
              (KNOTENLISTE_NEU
                      (PLAN-KNOTENLISTE
                               PLAN_NEU))
              (ERZEUGER_NEU
                      (KNOTEN-KOPIEREN
                               ERZEUGER_ALT
                               KNOTENLISTE_NEU))
              (ZERSTOERER_NEU
                      (KNOTEN-KOPIEREN
                               ZERSTOERER_ALT
                               KNOTENLISTE_NEU))
              (NUTZER_NEU
                      (KNOTEN-KOPIEREN
                               NUTZER_ALT
                               KNOTENLISTE_NEU))
              (WERTCONSTRAINT_NEU
                      (CONSTRAINT-EINBAU
                               ZERSTOERER_NEU
                               ZERSTOERER_EFFEKT
                               NUTZER_NEU
                               MERKMAL
                               PLAN_NEU)))
             (SETF (KNOTEN-NONCODESIGNATIONS ZERSTOERER_NEU)
                   (CONS
                       WERTCONSTRAINT_NEU
                       (KNOTEN-NONCODESIGNATIONS ZERSTOERER_NEU)))
             (SETF (KNOTEN-NONCODESIGNATIONS NUTZER_NEU)
                   (CONS
                       WERTCONSTRAINT_NEU
```

```
                        (KNOTEN-NONCODESIGNATIONS NUTZER_NEU)))
              (NUTZER-ABHAENGIGKEIT-EINTRAGEN
                       PLAN_NEU
                       NUTZER_NEU
                       MERKMAL
                       ERZEUGER_NEU)))

; -SEPARIEREN-----------------------------------------------------------



  (DEFUN SUCHE-ALTE-RETTER (PLANKNOTENLISTE
                            NUTZER
                            MERKMAL
                            ERZEUGER
                            ZERSTOERER)
; ---------------------------------------------------------------------
; Wert:                 - Falls Retter im gerade bearbeiteten Plan
;                         mit der Knotenliste PLANKNOTENLISTE exi-
;                         stent:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung (= 'ALT),
;                             2. Element: Retter(-Knoten), welcher
;                                         fuer den Nutzer(-Knoten) die
;                                         noch unerfuellte Vorbedin-
;                                         gung MERKMAL erzeugen koenn-
;                                         te,
;                             3. Element: Effekt (Nachbedingung des
;                                         Retters, welche mit der
;                                         Vorbedingung MERKMAL des
;                                         NUTZERs codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTENLISTE
;                       Knotenliste des gerade bearbeiteten Planes.
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des gerade be-
;                       arbeiteten Planes.
;
;                         MERKMAL
;                       Betrachtete Vorbedingung von NUTZER.
;
;                         ERZEUGER
;                       Zur Erzeugung von MERKMAL (-> NUTZER) bestimm-
;                       ter Knoten des gerade bearbeiteten Planes.
;
;                         ZERSTOERER
;                       Knoten des gerade bearbeiteten Planes, der die
;                       Abhaengigkeit ERZEUGER <-> NUTZER stoert.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             BESTIMME-ZIELKNOTEN
;                       BESTIMME-DIREKTE-VORGAENGER
;                       BESTIMME-VORGAENGER
;                       BESTIMME-STARTKNOTEN
```

```
;                       SUCHE-ALTE-RETTER-HILF
;
; aufgerufen von:       ERMITTLE-RETTER
;
; -SUCHE-ALTE-RETTER-------------------------------------------------

        (LET* ((ZIELKNOTEN
                    (BESTIMME-ZIELKNOTEN
                           PLANKNOTENLISTE))
               (GENERATION
                    (BESTIMME-DIREKTE-VORGAENGER
                           ZIELKNOTEN)))
              (SUCHE-ALTE-RETTER-HILF
                       GENERATION
                       (BESTIMME-DIREKTE-VORGAENGER
                              GENERATION)
                       NUTZER
                       MERKMAL
                       ERZEUGER
                       ZERSTOERER
                       (BESTIMME-VORGAENGER
                              ZERSTOERER)
                       NIL                  ; Initialwert fuer
                                            ; K_R_M_TRIPELLISTE.
                       (LIST
                           (BESTIMME-STARTKNOTEN
                                  PLANKNOTENLISTE))) ))

; -SUCHE-ALTE-RETTER-------------------------------------------------



  (DEFUN SUCHE-ALTE-RETTER-HILF (GENERATION
                                 NAECHSTE_GENERATION
                                 NUTZER
                                 MERKMAL
                                 ERZEUGER
                                 ZERSTOERER
                                 ZERSTOERER_VORGAENGER
                                 K_R_M_TRIPELLISTE
                                 MARKIERTE_KNOTEN)
; ------------------------------------------------------------------
; Wert:                 - Falls Retter im gerade bearbeiteten Plan
;                         mit der Knotenliste PLANKNOTENLISTE exi-
;                         stent:
;                           Liste dreielementiger Unterlisten mit fol-
;                           gendem Aufbau:
;                             1. Element: Kennung (= 'ALT),
;                             2. Element: Retter(-Knoten), welcher
;                                         fuer den Nutzer(-Knoten) die
;                                         noch unerfuellte Vorbedin-
;                                         gung MERKMAL erzeugen koenn-
;                                         te,
;                             3. Element: Effekt (nichtnegierte Nach-
;                                         bedingung des Retters-Kno-
;                                         tens (s.o.), welche mit der
;                                         Vorbedingung MERKMAL des
;                                         NUTZERs codesignierbar ist).
;                       - Sonst:
;                           NIL.
;
```

```
; Globale Variablen:    --
;
; Eingangsparameter:      GENERATION
;                       Aktuell betrachtete Knotengeneration bzw. Rest
;                       davon.
;
;                         NAECHSTE_GENERATION
;                       Knotengeneration direkt vor GENERATION (in
;                       Richtung Startknoten).
;
;                         NUTZER
;                       Gewaehlter ungeloester Knoten des gerade be-
;                       arbeiteten Planes.
;
;                         MERKMAL
;                       Betrachtete Vorbedingung von NUTZER.
;
;                         ERZEUGER
;                       Zur Erzeugung von MERKMAL (-> NUTZER) bestimm-
;                       ter Knoten des gerade bearbeiteten Planes.
;
;                         ZERSTOERER
;                       Knoten des gerade bearbeiteten Planes, der die
;                       Abhaengigkeit ERZEUGER <-> NUTZER stoert.
;
;                         ZERSTOERER_VORGAENGER
;                       Liste saemtlicher Vorgaenger-Knoten von ZER-
;                       STOERER im gerade bearbeiteten Plan.
;
;                         K_R_M_TRIPELLISTE
;                       Resultatliste.
;
;                         MARKIERTE_KNOTEN
;                       Liste der bereits untersuchten Knoten.
;
; Nebeneffekte:         Erzeugen der Resultatliste.
;
; ruft auf:             sich selbst
;                       BESTIMME-DIREKTE-VORGAENGER
;                       IST-VORGAENGER
;                       IST-ERZEUGER
;
; aufgerufen von:       sich selbst
;                       SUCHE-ALTE-RETTER
;
; -SUCHE-ALTE-RETTER-HILF-----------------------------------------------

        (COND ((AND
                   (NULL GENERATION)
                   (NULL NAECHSTE_GENERATION))
               K_R_M_TRIPELLISTE)
              ((NULL GENERATION)
               (LET ((KANDIDATEN
                          (REMOVE-IF
                              #'(LAMBDA (PLANKNOTEN)
                                     (MEMBER PLANKNOTEN
                                             MARKIERTE_KNOTEN))
                              NAECHSTE_GENERATION)))
                 (SUCHE-ALTE-RETTER-HILF
                     KANDIDATEN
                     (BESTIMME-DIREKTE-VORGAENGER
                            KANDIDATEN)
```

```
                        NUTZER
                        MERKMAL
                        ERZEUGER
                        ZERSTOERER
                        ZERSTOERER_VORGAENGER
                        K_R_M_TRIPELLISTE
                        (APPEND
                             KANDIDATEN
                             MARKIERTE_KNOTEN))) )
               ( T  (LET ((BETRACHTETER_KNOTEN
                              (CAR GENERATION)))
                         (SUCHE-ALTE-RETTER-HILF
                            (CDR GENERATION)
                            NAECHSTE_GENERATION
                            NUTZER
                            MERKMAL
                            ERZEUGER
                            ZERSTOERER
                            ZERSTOERER_VORGAENGER
                            (APPEND
                               (AND
                                  (NOT (EQL BETRACHTETER_KNOTEN
                                            ZERSTOERER))
                                  (NOT (EQL BETRACHTETER_KNOTEN
                                            NUTZER))
                                  (NOT (EQL BETRACHTETER_KNOTEN
                                            ERZEUGER))
                                  (NOT (IST-VORGAENGER
                                            BETRACHTETER_KNOTEN
                                            ZERSTOERER
                                            ZERSTOERER_VORGAENGER))
                                  (NOT (IST-VORGAENGER
                                            NUTZER
                                            BETRACHTETER_KNOTEN))
                                  (IST-ERZEUGER
                                         BETRACHTETER_KNOTEN
                                         NUTZER
                                         MERKMAL))
                               K_R_M_TRIPELLISTE)
                            MARKIERTE_KNOTEN))) ))

; -SUCHE-ALTE-RETTER-HILF-----------------------------------------



  (DEFUN SUCHE-ZERSTOERER (PLANKNOTENLISTE
                           NUTZER
                           MERKMAL
                           ERZEUGER)
; ---------------------------------------------------------------------
; Wert:                 - Falls (mindestens eine) Konflikt-Situation
;                         innerhalb des gerade bearbeiteten Plans ge-
;                         geben:
;                           Zweielementige Liste mit folgendem Aufbau:
;                             1. Element: Zerstoerer-Knoten,
;                             2. Element: Zerstoerer-Effekt
;                                         (= Nichtnegierter Part des
;                                            negierten Nachbedingungs-
;                                            Merkmals des Zerstoerer-
;                                            Knotens, der mit der
;                                            unerfuellten Vorbedingung
```

```
;                                          MERKMAL (-> NUTZER) code-
;                                          signierbar ist).

;                      (Anm.:  Sind mehrere Zerstoerer vorhanden, so
;                              wird der dem Zielknoten des betrachte-
;                              ten Plans am naechsten gelegene zusam-
;                              men mit seinem Effekt als Wert gelie-
;                              fert!)
;
;                      - Sonst:
;                          NIL.
;
; Globale Variablen:   --
;
; Eingangsparameter:      PLANKNOTENLISTE
;                      Knotenliste des gerade bearbeiteten Planes.
;
;                        NUTZER
;                      Gewaehlter ungeloester Knoten des gerade be-
;                      arbeiteten Planes.
;
;                        MERKMAL
;                      Betrachtete Vorbedingung von NUTZER.
;
;                        ERZEUGER
;                      Zur Erzeugung von MERKMAL (-> NUTZER) bestimm-
;                      ter Knoten des gerade bearbeiteten Planes.
;
; Nebeneffekte:        Erzeugen der Resultatliste.
;
; ruft auf:            BESTIMME-DIREKTE-VORGAENGER
;                      BESTIMME-ZIELKNOTEN
;                      BESTIMME-VORGAENGER
;                      SUCHE-ZERSTOERER-HILF
;
; aufgerufen von:      KONFLIKT-LOESUNGEN
;                      NUTZER-ABHAENGIGKEIT-EINTRAGEN
;
; -SUCHE-ZERSTOERER-----------------------------------------------------

       (LET ((GENERATION
                  (BESTIMME-DIREKTE-VORGAENGER
                         (BESTIMME-ZIELKNOTEN
                                PLANKNOTENLISTE))) )
            (SUCHE-ZERSTOERER-HILF
                  GENERATION
                  (BESTIMME-DIREKTE-VORGAENGER
                         GENERATION)
                  NUTZER
                  MERKMAL
                  ERZEUGER
                  (BESTIMME-VORGAENGER
                         ERZEUGER))) )

; -SUCHE-ZERSTOERER-----------------------------------------------------



  (DEFUN SUCHE-ZERSTOERER-HILF (GENERATION
                                NAECHSTE_GENERATION
                                NUTZER
```

```
                                MERKMAL
                                ERZEUGER
                                ERZEUGER_VORGAENGER)
; ---------------------------------------------------------------------
; Wert:                  - Falls (mindestens eine) Konflikt-Situation
;                          innerhalb des gerade bearbeiteten Plans ge-
;                          geben:
;                            Zweielementige Liste mit folgendem Aufbau:
;                              1. Element: Zerstoerer-Knoten,
;                              2. Element: Zerstoerer-Effekt
;                                        (= Nichtnegierter Part des
;                                           negierten Nachbedingungs-
;                                           Merkmals des Zerstoerer-
;                                           Knotens, der mit der
;                                           unerfuellten Vorbedingung
;                                           MERKMAL (-> NUTZER) code-
;                                           signierbar ist).
;                        - Sonst:
;                            NIL.
;
; Globale Variablen:     --
;
; Eingangsparameter:       GENERATION
;                        Aktuell betrachtete Knotengeneration bzw. Rest
;                        davon.
;
;                          NAECHSTE_GENERATION
;                        Knotengeneration direkt vor GENERATION (in
;                        Richtung Startknoten).
;
;                          NUTZER
;                        Gewaehlter ungeloester Knoten des gerade be-
;                        arbeiteten Planes.
;
;                          MERKMAL
;                        Betrachtete Vorbedingung von NUTZER.
;
;                          ERZEUGER
;                        Zur Erzeugung von MERKMAL (-> NUTZER) bestimm-
;                        ter Knoten des gerade bearbeiteten Planes.
;
;                          ERZEUGER_VORGAENGER
;                        Liste saemtlicher Vorgaenger-Knoten von ERZEU-
;                        GER im gerade bearbeiten Plan.
;
; Nebeneffekte:          Erzeugen der Resultatliste.
;
; ruft auf:              sich selbst
;                        BESTIMME-DIREKTE-VORGAENGER
;                        IST-VORGAENGER
;                        IST-ZERSTOERER
;
; aufgerufen von:        sich selbst
;                        SUCHE-ZERSTOERER
;
; -SUCHE-ZERSTOERER-HILF-----------------------------------------------

        (COND ((AND
                   (NULL GENERATION)
                   (NULL NAECHSTE_GENERATION))
               NIL)
              ((NULL GENERATION)
```

```
                (SUCHE-ZERSTOERER-HILF
                       NAECHSTE_GENERATION
                       (BESTIMME-DIREKTE-VORGAENGER
                              NAECHSTE_GENERATION)
                       NUTZER
                       MERKMAL
                       ERZEUGER
                       ERZEUGER_VORGAENGER))
               ( T  (LET ((BETRACHTETER_KNOTEN
                                 (CAR GENERATION)))
                         (OR
                            (AND
                               (NOT (EQL BETRACHTETER_KNOTEN
                                         NUTZER))
                               (NOT (EQL BETRACHTETER_KNOTEN
                                         ERZEUGER))
                               (NOT (IST-VORGAENGER
                                           BETRACHTETER_KNOTEN
                                           ERZEUGER
                                           ERZEUGER_VORGAENGER))
                               (NOT (IST-VORGAENGER
                                           NUTZER
                                           BETRACHTETER_KNOTEN))
                               (IST-ZERSTOERER
                                        BETRACHTETER_KNOTEN
                                        NUTZER
                                        MERKMAL))
                            (SUCHE-ZERSTOERER-HILF
                                   (CDR GENERATION)
                                   NAECHSTE_GENERATION
                                   NUTZER
                                   MERKMAL
                                   ERZEUGER
                                   ERZEUGER_VORGAENGER))) )))

; -SUCHE-ZERSTOERER-HILF---------------------------------------------


; =FUNKTIONEN ZUR KONFLIKT-BEHEBUNG==================================
```

```
; ======================================================================
; =============    FUNKTIONEN ZUR PLAN-DUPLIZIERUNG    ================
; ======================================================================

; Stand: 29.05.1990


  (DEFUN CONSTRAINT-KOPIEREN (CONSTRAINT)
; ----------------------------------------------------------------------
; Wert:                 Neu erzeugtes Wertconstraint, dessen Eintraege
;                       WERTCONSTRAINT-NAME und -MERKMALPAAR mit denen
;                       von CONSTRAINT identisch sind. Der Eintrag
;                       WERTCONSTRAINT-KNOTEN des neu erzeugten Wert-
;                       constraints ist noch NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      CONSTRAINT
;                       Zu kopierendes Wertconstraint.
;
; Nebeneffekte:         Erzeugen einer neuen Wertconstraint-Struktur.
;
; ruft auf:             --
;
; aufgerufen von:       CONSTRAINTLISTE-KOPIEREN
;
; -CONSTRAINT-KOPIEREN-----------------------------------------------

        (MAKE-WERTCONSTRAINT
                :NAME
                    (WERTCONSTRAINT-NAME
                              CONSTRAINT)
                :MERKMALPAAR
                    (WERTCONSTRAINT-MERKMALPAAR
                              CONSTRAINT)))

; -CONSTRAINT-KOPIEREN-----------------------------------------------



  (DEFUN CONSTRAINTLISTE-KOPIEREN (C_LISTE)
; ----------------------------------------------------------------------
; Wert:                 Liste von Constraints, welche bis auf den Ein-
;                       trag mit Verweisen auf andere Knoten (also bis
;                       auf WERTCONSTRAINT-KNOTEN) eine Kopie (mit ei-
;                       genem Speicherbereich) der uebergebenen Wert-
;                       constraints in C_LISTE darstellen.
;
; Globale Variablen:    --
;
; Eingangsparameter:      C_LISTE
;                       Liste der zu kopierenden Wertconstraints.
;
; Nebeneffekte:         Erzeugen neuer Wertconstraint-Strukturen.
;
; ruft auf:             sich selbst
;                       CONSTRAINT-KOPIEREN
;
; aufgerufen von:       sich selbst
;                       PLAN-KOPIEREN
```

```
;
; -CONSTRAINTLISTE-KOPIEREN-----------------------------------------

        (COND ((NULL C_LISTE) NIL)
              ( T  (CONS
                       (CONSTRAINT-KOPIEREN
                                (CAR C_LISTE))
                       (CONSTRAINTLISTE-KOPIEREN
                                (CDR C_LISTE))) )))

; -CONSTRAINTLISTE-KOPIEREN-----------------------------------------



  (DEFUN KNOTEN-KOPIEREN (PLANKNOTEN
                          &OPTIONAL NEUE_KNOTENLISTE)
; ------------------------------------------------------------------
; Wert:                 - Falls NEUE_KNOTENLISTE <> NIL :
;                         Knoten aus NEUE_KNOTENLISTE, dessen Name mit
;                         dem von PLANKNOTEN identisch ist, oder NIL
;                         bei Nichtexistenz eines derartigen Knotens.
;                       - Falls NEUE_KNOTENLISTE = NIL :
;                         Neu erzeugter Knoten, dessen Eintraege KNO-
;                         TEN-NAME, -VARIABLEN und -OPERATOR mit denen
;                         von PLANKNOTEN identisch sind. Die uebrigen
;                         Eintraege sind NIL.
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTEN
;                       Zu bestimmender bzw. zu kopierender Plankno-
;                       ten.
;
;                         NEUE_KNOTENLISTE (optional!)
;                       Bei Angabe dieses optionalen Parameters im
;                       Funktionsaufruf wird PLANKNOTEN nicht durch
;                       Erzeugen einer neuen Knoten-Struktur mit ent-
;                       sprechenden Initialisierungen kopiert, sondern
;                       ein Verweis auf einen Knoten aus NEUE_KNOTEN-
;                       LISTE mit demselben Namen wie PLANKNOTEN (oder
;                       NIL bei Nichtexistenz eines derartigen Kno-
;                       tens) als Wert geliefert.
;
; Nebeneffekte:         Erzeugen einer neuen Knoten-Struktur, falls
;                       NEUE_KNOTENLISTE = NIL.
;
; ruft auf:             OBJEKT-BESTIMMUNG
;
; aufgerufen von:       ERZEUGER-ERFUELLT-MERKMAL
;                       ORDNUNG-VERSCHAERFEN
;                       RETTER-NEUTRALISIERT-ZERSTOERER
;                       SEPARIEREN
;                       KNOTENLISTE-KOPIEREN
;
; -KNOTEN-KOPIEREN--------------------------------------------------

        (COND ((NOT (NULL NEUE_KNOTENLISTE))
               (OBJEKT-BESTIMMUNG
                        (KNOTEN-NAME PLANKNOTEN)
                        'KNOTEN-NAME
                        NEUE_KNOTENLISTE))
```

```
              ( T  (MAKE-KNOTEN
                         :NAME
                             (KNOTEN-NAME PLANKNOTEN)
                         :VARIABLEN
                             (KNOTEN-VARIABLEN PLANKNOTEN)
                         :OPERATOR
                             (KNOTEN-OPERATOR PLANKNOTEN))) ))

; -KNOTEN-KOPIEREN-----------------------------------------------------



  (DEFUN KNOTENLISTE-KOPIEREN (K_LISTE)
; ---------------------------------------------------------------------
; Wert:                 Liste von Knoten, welche bis auf die Eintraege
;                       mit Verweisen auf andere Knoten oder Wertcon-
;                       straints (also bis auf KNOTEN-CODESIGNATIONS,
;                       -NONCODESIGNATIONS, -VORGAENGER, -ist_abhaen-
;                       gig_von und -erzeugt_fuer) eine Kopie (mit ei-
;                       genem Speicherbereich) der uebergebenen Knoten
;                       in K_LISTE darstellen.
;
; Globale Variablen:    --
;
; Eingangsparameter:      K_LISTE
;                       Liste der zu kopierenden Knoten.
;
; Nebeneffekte:         Erzeugen neuer Knoten-Strukturen.
;
; ruft auf:             sich selbst
;                       KNOTEN-KOPIEREN
;
; aufgerufen von:       sich selbst
;                       PLAN-KOPIEREN
;
; -KNOTENLISTE-KOPIEREN------------------------------------------------

        (COND ((NULL K_LISTE) NIL)
              ( T  (CONS
                       (KNOTEN-KOPIEREN
                             (CAR K_LISTE))
                       (KNOTENLISTE-KOPIEREN
                             (CDR K_LISTE))) )))

; -KNOTENLISTE-KOPIEREN------------------------------------------------



  (DEFUN PLAN-KOPIEREN (ORIGINALPLAN)
; ---------------------------------------------------------------------
; Wert:                 Kopie von ORIGINALPLAN, wobei gilt:
;                       saemtliche Wertconstraints und Knoten von ORI-
;                       GINALPLAN saemt saemtlicher Eintraege sind ko-
;                       piert mit eigenem internen Speicherbereich.
;                       Lediglich die Operator-Strukturen sind allen
;                       Plaenen gemeinsam. Verweise innerhalb von ORI-
;                       GINALPLAN auf Knoten oder Wertconstraints des
;                       ORIGINALPLANs werden zu internen Verweisen in-
;                       nerhalb der Plankopie transformiert.
;
```

```
; Globale Variablen:    --
;
; Eingangsparameter:      ORIGINALPLAN
;                       Plan, der kopiert werden und dabei unveraendert
;                       bleiben soll.
;
; Nebeneffekte:         Erzeugen einer Plankopie von ORIGINALPLAN.
;
; ruft auf:             KNOTENLISTE-KOPIEREN
;                       CONSTRAINTLISTE-KOPIEREN
;                       VERWEISE-HERSTELLEN
;
; aufgerufen von:       ERZEUGER-ERFUELLT-MERKMAL
;                       ORDNUNG-VERSCHAERFEN
;                       RETTER-NEUTRALISIERT-ZERSTOERER
;                       SEPARIEREN
;                       PLAN-AUSGEBEN
;
; -PLAN-KOPIEREN-----------------------------------------------------

        (LET* ((ALTE_KNOTENLISTE
                       (PLAN-KNOTENLISTE
                               ORIGINALPLAN))
               (ALTE_CONSTRAINTLISTE
                       (PLAN-WERTCONSTRAINTLISTE
                               ORIGINALPLAN))
               (NEUE_KNOTENLISTE
                       (KNOTENLISTE-KOPIEREN
                               ALTE_KNOTENLISTE))
               (NEUE_CONSTRAINTLISTE
                       (CONSTRAINTLISTE-KOPIEREN
                               ALTE_CONSTRAINTLISTE)))
              (VERWEISE-HERSTELLEN
                       NEUE_KNOTENLISTE
                       NEUE_CONSTRAINTLISTE
                       ALTE_KNOTENLISTE
                       ALTE_CONSTRAINTLISTE)
              (MAKE-PLAN
                       :KNOTENLISTE
                              NEUE_KNOTENLISTE
                       :WERTCONSTRAINTLISTE
                              NEUE_CONSTRAINTLISTE)))
; -PLAN-KOPIEREN-----------------------------------------------------



  (DEFUN VERWEISE-HERSTELLEN (NEUE_KNOTENLISTE
                              NEUE_CONSTRAINTLISTE
                              ALTE_KNOTENLISTE
                              ALTE_CONSTRAINTLISTE)
; ------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      NEUE_KNOTENLISTE
;                       Knotenliste des neuen ("frisch" kopierten)
;                       Planes. Ausser Namen, Variablen und Operator
;                       sind die Eintraege der Knoten aus NEUE_KNOTEN-
;                       LISTE eingangs noch mit NIL initialisiert.
;
```

```
;                       NEUE_CONSTRAINTLISTE
;                     Wertconstraintliste des neuen ("frisch" ko-
;                     pierten) Planes. Der Knoten-Eintrag der Wert-
;                     constraints aus NEUE_CONSTRAINTLISTE ist ein-
;                     gangs noch mit NIL initialisiert.
;
;                       ALTE_KNOTENLISTE
;                     Knotenliste des alten Planes, den es zu kopie-
;                     ren gilt.
;
;                       ALTE_CONSTRAINTLISTE
;                     Wertconstraintliste des alten Planes, den es
;                     zu kopieren gilt.
;
; Nebeneffekte:       Herstellen und Eintragen saemtlicher Verweise
;                     auf Knoten und Wertconstraints innerhalb des
;                     neuen ("frisch" kopierten) Planes.
;
; ruft auf:           VERWEISE-KOPIEREN
;                     VERWEISE-MIT-MERKMAL-KOPIEREN
;
; aufgerufen von:     PLAN-KOPIEREN
;
; -VERWEISE-HERSTELLEN-----------------------------------------------

      (DO* ((NEUE_KNOTEN      NEUE_KNOTENLISTE
                              (CDR NEUE_KNOTEN))
            (NEUER_KNOTEN     (CAR NEUE_KNOTENLISTE)
                              (CAR NEUE_KNOTEN))
            (ALTE_KNOTEN      ALTE_KNOTENLISTE
                              (CDR ALTE_KNOTEN))
            (ALTER_KNOTEN     (CAR ALTE_KNOTENLISTE)
                              (CAR ALTE_KNOTEN)))
           ((NULL NEUE_KNOTEN))
           (SETF (KNOTEN-CODESIGNATIONS NEUER_KNOTEN)
                 (VERWEISE-KOPIEREN
                       (KNOTEN-CODESIGNATIONS ALTER_KNOTEN)
                       'WERTCONSTRAINT-NAME
                       NEUE_CONSTRAINTLISTE))
           (SETF (KNOTEN-NONCODESIGNATIONS NEUER_KNOTEN)
                 (VERWEISE-KOPIEREN
                       (KNOTEN-NONCODESIGNATIONS ALTER_KNOTEN)
                       'WERTCONSTRAINT-NAME
                       NEUE_CONSTRAINTLISTE))
           (SETF (KNOTEN-VORGAENGER NEUER_KNOTEN)
                 (VERWEISE-KOPIEREN
                       (KNOTEN-VORGAENGER ALTER_KNOTEN)
                       'KNOTEN-NAME
                       NEUE_KNOTENLISTE))
           (SETF (KNOTEN-ist_abhaengig_von NEUER_KNOTEN)
                 (VERWEISE-MIT-MERKMAL-KOPIEREN
                       (KNOTEN-ist_abhaengig_von ALTER_KNOTEN)
                       NEUE_KNOTENLISTE))
           (SETF (KNOTEN-erzeugt_fuer NEUER_KNOTEN)
                 (VERWEISE-MIT-MERKMAL-KOPIEREN
                       (KNOTEN-erzeugt_fuer ALTER_KNOTEN)
                       NEUE_KNOTENLISTE)))
      (DO* ((NEUE_CONSTRAINTS   NEUE_CONSTRAINTLISTE
                                (CDR NEUE_CONSTRAINTS))
            (NEUES_CONSTRAINT   (CAR NEUE_CONSTRAINTLISTE)
                                (CAR NEUE_CONSTRAINTS))
            (ALTE_CONSTRAINTS   ALTE_CONSTRAINTLISTE
```

```
                                (CDR ALTE_CONSTRAINTS))
               (ALTES_CONSTRAINT   (CAR ALTE_CONSTRAINTLISTE)
                                (CAR ALTE_CONSTRAINTS)))
              ((NULL NEUE_CONSTRAINTS))
              (SETF (WERTCONSTRAINT-KNOTEN NEUES_CONSTRAINT)
                    (VERWEISE-KOPIEREN
                           (WERTCONSTRAINT-KNOTEN ALTES_CONSTRAINT)
                           'KNOTEN-NAME
                           NEUE_KNOTENLISTE))) )

; -VERWEISE-HERSTELLEN-----------------------------------------------



  (DEFUN VERWEISE-KOPIEREN (ALTE_VERWEISE
                            NAMETYP_VERWEISZIEL
                            NEUE_VERWEISZIELE)
; ---------------------------------------------------------------------
; Wert:                 - Falls NAMETYP_VERWEISZIEL = 'KNOTEN-NAME :
;                         Liste von Knoten, auf die im neuen (kopier-
;                         ten) Plan innerhalb eines Knoten-Eintrags
;                         (nur beim Eintrag KNOTEN-VORGAENGER) oder
;                         Wertconstraint-Eintrags (nur beim Eintrag
;                         WERTCONSTRAINT-KNOTEN) verwiesen werden
;                         soll, wobei als Vorlage die Verweise des
;                         entsprechenden Eintrags im alten (zu kopie-
;                         renden) Plan dienen.
;                       - Falls NAMETYP_VERWEISZIEL = 'WERTCONSTRAINT-
;                                                              NAME :
;                         Liste von Wertconstraints, auf die im neuen
;                         (kopierten) Plan innerhalb eines Knoten-Ein-
;                         trags (nur bei den Eintraegen KNOTEN-CODESIG-
;                         NATIONS und -NONCODESIGNATIONS) verwiesen
;                         werden soll, wobei als Vorlage die Verweise
;                         des entsprechenden Eintrags im alten (zu ko-
;                         pierenden) Plan dienen.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ALTE_VERWEISE
;                       Liste von Knoten oder Wertconstraints, welche
;                       den Eintrag eines Knotens (nur bei den Ein-
;                       traegen KNOTEN-VORGAENGER, -CODESIGNATIONS und
;                       -NONCODESIGNATIONS) oder eines Wertconstraints
;                       (nur bei dem Eintrag WERTCONSTRAINT-KNOTEN) im
;                       alten (zu kopierenden) Plan darstellen.
;
;                         NAMETYP_VERWEISZIEL
;                       Indikator fuer das Verweisziel (= 'KNOTEN-NAME
;                       oder 'WERTCONSTRAINT-NAME).
;
;                         NEUE_VERWEISZIELE
;                       Knoten- oder Wertconstraintliste des neuen ko-
;                       pierten Planes, welche die neuen Verweisziele
;                       bereitstellt.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       OBJEKT-BESTIMMUNG
;
```

```
; aufgerufen von:       sich selbst
;                       VERWEISE-HERSTELLEN
;
; -VERWEISE-KOPIEREN-----------------------------------------------

        (COND ((NULL ALTE_VERWEISE) NIL)
              ( T  (CONS
                       (OBJEKT-BESTIMMUNG
                                (FUNCALL NAMETYP_VERWEISZIEL
                                         (CAR ALTE_VERWEISE))
                                NAMETYP_VERWEISZIEL
                                NEUE_VERWEISZIELE)
                       (VERWEISE-KOPIEREN
                                (CDR ALTE_VERWEISE)
                                NAMETYP_VERWEISZIEL
                                NEUE_VERWEISZIELE))) ))

; -VERWEISE-KOPIEREN-----------------------------------------------



  (DEFUN VERWEISE-MIT-MERKMAL-KOPIEREN (ALTE_EINTRAEGE
                                        NEUE_VERWEISZIELE)
; ---------------------------------------------------------------------
; Wert:                 Liste von Knoten-Merkmal-Unterlisten, auf die
;                       im neuen (kopierten) Plan innerhalb eines Kno-
;                       ten-Eintrags (nur bei den Eintraegen KNOTEN-
;                       ist_abhaengig_von und erzeugt_fuer) verwiesen
;                       werden soll, wobei als Vorlage die Verweise
;                       des entsprechenden Eintrags im alten (zu ko-
;                       pierenden) Plans dienen.
;
; Globale Variablen:    --
;
; Eingangsparameter:       ALTE_EINTRAEGE
;                       Liste von Knoten-Merkmal-Unterlisten, welche
;                       den Eintrag eines Knotens (nur bei den Ein-
;                       traegen KNOTEN-ist_abhaengig_von und -erzeugt_
;                       fuer) im alten (zu kopierenden) Plan darstel-
;                       len.
;
;                         NEUE_VERWEISZIELE
;                       Knotenliste des neuen kopierten Planes, welche
;                       die neuen Knoten-Verweisziele bereitstellt.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       OBJEKT-BESTIMMUNG
;
; aufgerufen von:       sich selbst
;                       VERWEISE-HERSTELLEN
;
; -VERWEISE-MIT-MERKMAL-KOPIEREN-----------------------------------

        (COND ((NULL ALTE_EINTRAEGE) NIL)
              ( T  (LET* ((EINTRAG
                                (CAR ALTE_EINTRAEGE))
                          (KOPIERTER_EINTRAG
                                (COND ((= (LENGTH EINTRAG)
                                          3)
```

```
                                    (CONS
                                        (CAR EINTRAG)
                                        (CONS
                                            (OBJEKT-BESTIMMUNG
                                                (KNOTEN-NAME
                                                     (CADR EINTRAG))
                                                'KNOTEN-NAME
                                                NEUE_VERWEISZIELE)
                                            (CDDR EINTRAG))) )
                                   ( T
                                    (CONS
                                        (OBJEKT-BESTIMMUNG
                                            (KNOTEN-NAME
                                                 (CAR EINTRAG))
                                            'KNOTEN-NAME
                                            NEUE_VERWEISZIELE)
                                        (CDR EINTRAG))) )))
                           (CONS
                               KOPIERTER_EINTRAG
                               (VERWEISE-MIT-MERKMAL-KOPIEREN
                                     (CDR ALTE_EINTRAEGE)
                                     NEUE_VERWEISZIELE))) )))

; -VERWEISE-MIT-MERKMAL-KOPIEREN-------------------------------------


; =FUNKTIONEN ZUR PLAN-DUPLIZIERUNG==================================
```

```
; ======================================================================
; =============      FUNKTIONEN ZUR PLAN-AUSGABE       ===============
; ======================================================================

; Stand: 29.05.1990



  (DEFUN BESTIMME-MERKMALLAENGE (MERKMAL)
; ----------------------------------------------------------------------
; Wert:                 Zahl der Zeichen, aus denen sich MERKMAL zu-
;                       sammensetzt.
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMAL
;                       Merkmal, dessen Zeichenzahl zu bestimmen ist.
;
; Nebeneffekte:         --
;
; ruft auf:             sich selbst
;                       BESTIMME-STRINGLAENGE
;                       IST-NEGIERT
;
; aufgerufen von:       sich selbst
;                       MERKMALBINDUNGEN-AUSGEBEN
;
; -BESTIMME-MERKMALLAENGE-----------------------------------------

        (COND ((NULL MERKMAL) 1)
              ( T  (+ (BESTIMME-STRINGLAENGE
                                (CAR MERKMAL))
                       (COND ((IST-NEGIERT MERKMAL)
                              (+ 4
                                 (BESTIMME-MERKMALLAENGE
                                       (CADR MERKMAL))) )
                             ( T
                              (+ 1
                                 (BESTIMME-MERKMALLAENGE
                                       (CDR MERKMAL))) ))) )))

; -BESTIMME-MERKMALLAENGE-----------------------------------------



  (DEFUN BESTIMME-STRINGLAENGE (ZEICHENFOLGE)
; ----------------------------------------------------------------------
; Wert:                 Zahl der Zeichen (characters), aus denen sich
;                       ZEICHENFOLGE zusammensetzt.
;
; Globale Variablen:    --
;
; Eingangsparameter:      ZEICHENFOLGE
;                       Symbolisches oder numerisches Atom, fuer wel-
;                       ches die Character-Zahl seiner externen Reprae-
;                       sentation bestimmt werden soll.
;
; Nebeneffekte:         --
;
```

```
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       BESTIMME-MERKMALLAENGE
;                       KNOTEN-AUSGEBEN
;
; -BESTIMME-STRINGLAENGE-----------------------------------------------

        (COND ((SYMBOLP ZEICHENFOLGE)
               (LENGTH (SYMBOL-NAME
                           ZEICHENFOLGE)))
              ((AND
                  (NUMBERP ZEICHENFOLGE)
                  (NOT (ZEROP ZEICHENFOLGE)))
               (+ 1
                  (BESTIMME-STRINGLAENGE
                        (TRUNCATE ZEICHENFOLGE
                                  10))) )
              ( T  0)))

; -BESTIMME-STRINGLAENGE-----------------------------------------------



  (DEFUN BEZIEHUNGEN-AUSGEBEN (KMPAAR_LISTE
                               PRAEPOSITION)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      KMPAAR_LISTE
;                       Knoten-Merkmal-Paare-Liste (stammend aus einem
;                       Knoten-Eintrag KNOTEN-ist_abhaengig_von oder
;                       KNOTEN-erzeugt_fuer), die ausgegeben werden
;                       soll.
;
;                         PRAEPOSITION
;                       = 'VON  (falls Abhaengigkeiten ausgegeben wer-
;                                den sollen),
;                       = 'FUER (falls Merkmal-Erzeugungen ausgegeben
;                                den sollen).
;                       Dient lediglich der verstaendlicheren Ausgabe.
;
; Nebeneffekte:         Formatierte Ausgabe von KAPAAR_LISTE.
;
; ruft auf:             sich selbst
;                       BLANKS
;
; aufgerufen von:       sich selbst
;                       KNOTEN-AUSGEBEN
;
; -BEZIEHUNGEN-AUSGEBEN------------------------------------------------

        (COND ((NULL KMPAAR_LISTE))
              ( T  (LET ((KMPAAR
                               (CAR KMPAAR_LISTE))
                         (REST
                               (CDR KMPAAR_LISTE)))
                        (PRINC (CADR KMPAAR))
                        (BLANKS 2)
```

```
                        (PRINC PRAEPOSITION)
                        (PRINC '|  KNOTEN-|)
                        (PRINC (KNOTEN-NAME
                                    (CAR KMPAAR)))
                        (COND ((NOT (NULL REST))
                               (TERPRI)
                               (BLANKS 20)
                               (BEZIEHUNGEN-AUSGEBEN
                                      REST
                                      PRAEPOSITION))) ))) )

; -BEZIEHUNGEN-AUSGEBEN-----------------------------------------------



  (DEFUN BLANKS (ANZAHL)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      ANZAHL
;                       Zahl der auszugebenden Leerzeichen.
;
; Nebeneffekte:         Ausgabe von ANZAHL Leerzeichen ohne vorange-
;                       hendem oder nachfolgendem Zeilensprung.
;
; ruft auf:             --
;
; aufgerufen von:       BEZIEHUNGEN-AUSGEBEN
;                       KAPAARE-AUSGEBEN
;                       KMPAARE-AUSGEBEN
;                       KNOTEN-AUSGEBEN
;                       MERKMALBINDUNGEN-AUSGEBEN
;                       MERKMALE-AUSGEBEN
;                       TEXT-AUSGABE1
;                       VARIABLENBINDUNGEN-AUSGEBEN
;                       VORGAENGER-AUSGEBEN
;
; -BLANKS-------------------------------------------------------------

        (DO ((I     ANZAHL     (- I 1)))
            ((ZEROP I))
            (PRINC '| |)))

; -BLANKS-------------------------------------------------------------



  (DEFUN ERGEBNIS-AUSGABE (GELOESTER_PLAN)
; ---------------------------------------------------------------------
; Wert:                 'FERTIG
;
; Globale Variablen:    START
;                       ZIEL
;
; Eingangsparameter:      GELOESTER_PLAN
;                       Auszugebender geloester nichtlinearer Plan.
;
; Nebeneffekte:         Formatierte Ausgabe von GELOESTER_PLAN.
```

```
;
; ruft auf:             UNTERSTREICHEN
;                       MERKMALE-AUSGEBEN
;                       PLAN-AUSGEBEN
;
; aufgerufen von:       PLANEN
;
; -ERGEBNIS-AUSGABE-----------------------------------------------------

      (TERPRI)
      (UNTERSTREICHEN 68)
      (TERPRI)
      (PRINC '|GESTELLTES PLANUNGSPROBLEM:|)
      (TERPRI)
      (TERPRI)
      (UNTERSTREICHEN 68)
      (PRINC '|GEGEBENE STARTSITUATION:|)
      (TERPRI)
      (PRINC '|=======================|)
      (MERKMALE-AUSGEBEN
               (OPERATOR-NACHBEDINGUNGEN
                      START))
      (TERPRI)
      (PRINC '|ERWUENSCHTE FINALSITUATION:|)
      (TERPRI)
      (PRINC '|============================|)
      (MERKMALE-AUSGEBEN
               (OPERATOR-VORBEDINGUNGEN
                      ZIEL))
      (TERPRI)
      (UNTERSTREICHEN 68)
      (TERPRI)
      (PRINC '|NICHTLINEARER PLAN ZUR LOESUNG DES PROBLEMS:|)
      (TERPRI)
      (PLAN-AUSGEBEN
               GELOESTER_PLAN))

; -ERGEBNIS-AUSGABE-----------------------------------------------------



  (DEFUN KAPAARE-AUSGEBEN (KAPAAR_LISTE
                           &OPTIONAL (BLANK_ANZAHL 0))
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      KAPAAR_LISTE
;                       Knoten-Argument-Paare-Liste, die ausgegeben
;                       werden soll.
;
;                         BLANK_ANZAHL (optional!)
;                       Zahl auszugebender Leerzeichen unmittelbar
;                       nach jedem Zeilensprung.
;
; Nebeneffekte:         Formatierte Ausgabe von KAPAAR_LISTE.
;
; ruft auf:             sich selbst
;                       IST-VARIABLE
;                       BLANKS
```

```
;
; aufgerufen von:       sich selbst
;                       VARIABLENBINDUNGEN-AUSGEBEN
;
; -KAPAARE-AUSGEBEN-----------------------------------------------

        (COND ((NULL KAPAAR_LISTE))
              ( T  (LET* ((KAPAAR
                                (CAR KAPAAR_LISTE))
                          (REST
                                (CDR KAPAAR_LISTE))
                          (ARGUMENT
                                (CADR KAPAAR)))
                         (PRINC ARGUMENT)
                         (COND ((IST-VARIABLE ARGUMENT)
                                (TERPRI)
                                (BLANKS
                                     BLANK_ANZAHL)
                                (PRINC '| [KNOTEN-|)
                                (PRINC (KNOTEN-NAME
                                            (CAR KAPAAR)))
                                (PRINC '|]|)))
                         (COND ((NOT (NULL REST))
                                (TERPRI)
                                (BLANKS
                                     BLANK_ANZAHL)
                                (KAPAARE-AUSGEBEN
                                     REST
                                     BLANK_ANZAHL))) ))) )

; -KAPAARE-AUSGEBEN-----------------------------------------------



  (DEFUN KMPAARE-AUSGEBEN (KMPAAR_LISTE
                           BLANK_ANZAHL)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      KMPAAR_LISTE
;                       Knoten-Merkmal-Paare-Liste, die ausgegeben
;                       werden soll.
;
;                         BLANK_ANZAHL
;                       Zahl auszugebender Leerzeichen unmittelbar
;                       nach jedem Zeilensprung.
;
; Nebeneffekte:         Formatierte Ausgabe von KMPAAR_LISTE.
;
; ruft auf:             sich selbst
;                       BLANKS
;
; aufgerufen von:       sich selbst
;                       MERKMALBINDUNGEN-AUSGEBEN
;
; -KMPAARE-AUSGEBEN-----------------------------------------------

        (COND ((NULL KMPAAR_LISTE))
              ( T  (LET ((KMPAAR
```

```
                              (CAR KMPAAR_LISTE))
                         (REST
                              (CDR KMPAAR_LISTE)))
                        (PRINC (CADR KMPAAR))
                        (TERPRI)
                        (BLANKS
                             BLANK_ANZAHL)
                        (PRINC '| [KNOTEN-|)
                        (PRINC (KNOTEN-NAME
                                    (CAR KMPAAR)))
                        (PRINC '|]|)
                        (COND ((NOT (NULL REST))
                               (TERPRI)
                               (BLANKS
                                    BLANK_ANZAHL)
                               (KMPAARE-AUSGEBEN
                                    REST
                                    BLANK_ANZAHL))) ))) )

; -KMPAARE-AUSGEBEN-----------------------------------------------------



  (DEFUN KNOTEN-AUSGEBEN (K_LISTE)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      K_LISTE
;                       Entweder einzelner Planknoten oder Planknoten-
;                       liste, der oder die ausgegeben werden
;                       soll(en).
;
; Nebeneffekte:         Formatierte Ausgabe von K_LISTE.
;
; ruft auf:             sich selbst
;                       BLANKS
;                       UNTERSTREICHEN
;                       BESTIMME-STRINGLAENGE
;                       VARIABLENBINDUNGEN-AUSGEBEN
;                       MERKMALBINDUNGEN-AUSGEBEN
;                       VORGAENGER-AUSGEBEN
;                       BEZIEHUNGEN-AUSGEBEN
;                       BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
;
; aufgerufen von:       sich selbst
;                       PLAN-AUSGEBEN-HILF
;
; -KNOTEN-AUSGEBEN------------------------------------------------------

        (COND ((NULL K_LISTE)
               (TERPRI))
              ((ATOM K_LISTE)
               (KNOTEN-AUSGEBEN
                    (LIST K_LISTE)))
              ( T  (LET* ((PLANKNOTEN
                                (CAR K_LISTE))
                          (KNOTENNAME
                                (KNOTEN-NAME
                                     PLANKNOTEN))
```

```
 (VARIABLENLISTE
      (KNOTEN-VARIABLEN
           PLANKNOTEN))
 (OPERATORNAME
      (KNOTEN-OPERATOR
           PLANKNOTEN))
 (OPERATORSTRUKTUR
      (EVAL OPERATORNAME))
 (CODS
      (KNOTEN-CODESIGNATIONS
           PLANKNOTEN))
 (NONCODS
      (KNOTEN-NONCODESIGNATIONS
           PLANKNOTEN)))
(TERPRI)
(PRINC '|  KNOTEN-|)
(PRINC KNOTENNAME)
(PRINC '|:|)
(TERPRI)
(BLANKS 2)
(UNTERSTREICHEN
         (+ 8
            (BESTIMME-STRINGLAENGE
                    KNOTENNAME)))
(PRINC 'VARIABLEN)
(BLANKS 11)
(COND ((NOT (NULL VARIABLENLISTE))
       (PRINC VARIABLENLISTE)))
(VARIABLENBINDUNGEN-AUSGEBEN
         PLANKNOTEN
         VARIABLENLISTE)
(PRINT 'OPERATOR)
(BLANKS 11)
(PRINC OPERATORNAME)
(PRINT 'MERKMAL-BINDUNGEN)
(TERPRI)
(PRINC '| AN VORBEDINGUNGEN  |)
(MERKMALBINDUNGEN-AUSGEBEN
         PLANKNOTEN
         CODS
         NONCODS
         (OPERATOR-VORBEDINGUNGEN
               OPERATORSTRUKTUR))
(TERPRI)
(PRINC '| AN NACHBEDINGUNGEN |)
(MERKMALBINDUNGEN-AUSGEBEN
         PLANKNOTEN
         CODS
         NONCODS
         (OPERATOR-NACHBEDINGUNGEN
               OPERATORSTRUKTUR))
(PRINT 'VORGAENGER-KNOTEN)
(BLANKS 2)
(VORGAENGER-AUSGEBEN
         (KNOTEN-VORGAENGER
               PLANKNOTEN))
(PRINT 'ABHAENGIGKEITEN)
(BLANKS 4)
(BEZIEHUNGEN-AUSGEBEN
     (REVERSE
         (BESTIMME-ETABLIERTE-ABHAENGIGKEITEN
               PLANKNOTEN))
```

```
                              'VON)
                        (PRINT 'MERKMAL-ERZEUGUNGEN)
                        (BEZIEHUNGEN-AUSGEBEN
                                 (KNOTEN-erzeugt_fuer
                                       PLANKNOTEN)
                                 'FUER)
                        (TERPRI)
                        (KNOTEN-AUSGEBEN
                                 (CDR K_LISTE))) )))

; -KNOTEN-AUSGEBEN-----------------------------------------------------




  (DEFUN KONSTANTEN-AUSGEBEN (KONSTANTENLISTE
                              &OPTIONAL (BLANK_ANZAHL 0))
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      KONSTANTENLISTE
;                       Liste auszugebender Konstanten.
;
;                         BLANK_ANZAHL
;                       Zahl auszugebender Leerzeichen unmittelbar
;                       nach jedem Zeilensprung.
;
; Nebeneffekte:         Formatierte Ausgabe von KONSTANTENLISTE.
;
; ruft auf:             sich selbst
;                       BLANKS
;
; aufgerufen von:       sich selbst
;                       VARIABLENBINDUNGEN-AUSGEBEN
;
; -KONSTANTEN-AUSGEBEN-------------------------------------------------

        (COND ((NULL KONSTANTENLISTE))
              ( T  (LET* ((KONSTANTE
                                 (CAR KONSTANTENLISTE))
                          (REST
                                 (CDR KONSTANTENLISTE)))
                        (PRINC KONSTANTE)
                        (COND ((NOT (NULL REST))
                               (TERPRI)
                               (BLANKS
                                    BLANK_ANZAHL)
                               (KONSTANTEN-AUSGEBEN
                                    REST
                                    BLANK_ANZAHL))) ))) )

; -KONSTANTEN-AUSGEBEN-------------------------------------------------




  (DEFUN MERKMALBINDUNGEN-AUSGEBEN (PLANKNOTEN
                                    COD_CONSTRAINTS
                                    NONCOD_CONSTRAINTS
                                    MERKMALLISTE)
```

```
; ------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTEN
;                       Bezugs-Planknoten zu COD_CONSTRAINTS, NONCOD_
;                       CONSTRAINTS und MERKMALLISTE.
;
;                         COD_CONSTRAINTS
;                       Liste (saemtlicher) Codesignation-Wertcon-
;                       straints von PLANKNOTEN.
;
;                         NONCOD_CONSTRAINTS
;                       Liste (saemtlicher) Noncodesignation-Wertcon-
;                       straints von PLANKNOTEN.
;
;                         MERKMALLISTE
;                       Liste (saemtlicher) Vor- oder Nachbedingungen
;                       der Operatorinstanz PLANKNOTEN.
;
; Nebeneffekte:         Formatierte Ausgabe saemtlicher Vor- oder
;                       saemtlicher Nachbedingungen von PLANKNOTEN
;                       samt den per Wertconstraints direkt verbunde-
;                       nen Partner-Merkmalen.
;
; ruft auf:             sich selbst
;                       BESTIMME-MERKMALLAENGE
;                       CONSTRAINTS-ORDNEN
;                       BLANKS
;                       KMPAARE-AUSGEBEN
;
; aufgerufen von:       sich selbst
;                       KNOTEN-AUSGEBEN
;
; -MERKMALBINDUNGEN-AUSGEBEN---------------------------------------

       (COND ((NULL MERKMALLISTE))
             ( T  (LET* ((MERKMAL
                              (CAR MERKMALLISTE))
                         (REST
                              (CDR MERKMALLISTE))
                         (BLANK_ANZAHL
                              (+ 24
                                 (BESTIMME-MERKMALLAENGE
                                       MERKMAL)))
                         (COD_KM_RESTCL
                              (CONSTRAINTS-ORDNEN
                                       COD_CONSTRAINTS
                                       PLANKNOTEN
                                       MERKMAL))
                         (NONCOD_KM_RESTCL
                              (CONSTRAINTS-ORDNEN
                                       NONCOD_CONSTRAINTS
                                       PLANKNOTEN
                                       MERKMAL))
                         (COD_KMPAARE
                              (CAR COD_KM_RESTCL))
                         (NONCOD_KMPAARE
                              (CAR NONCOD_KM_RESTCL)))
                        (PRINC MERKMAL)
                        (COND ((NOT (NULL COD_KMPAARE))
```

```
                              (PRINC '| =  |)
                              (KMPAARE-AUSGEBEN
                                   COD_KMPAARE
                                   BLANK_ANZAHL)))
                         (COND ((NOT (NULL NONCOD_KMPAARE))
                              (COND ((NOT (NULL COD_KMPAARE))
                                     (TERPRI)
                                     (BLANKS
                                          (- BLANK_ANZAHL
                                             4))) )
                              (PRINC '| <> |)
                              (KMPAARE-AUSGEBEN
                                   NONCOD_KMPAARE
                                   BLANK_ANZAHL)))
                         (COND ((NOT (NULL REST))
                              (TERPRI)
                              (BLANKS 20)
                              (MERKMALBINDUNGEN-AUSGEBEN
                                   PLANKNOTEN
                                   (CADR COD_KM_RESTCL)
                                   (CADR NONCOD_KM_RESTCL)
                                   (CDR MERKMALLISTE))) ))) ))

; -MERKMALBINDUNGEN-AUSGEBEN----------------------------------------




  (DEFUN MERKMALE-AUSGEBEN (MERKMALLISTE)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      MERKMALLISTE
;                       Liste auszugebender Merkmale.
;
; Nebeneffekte:         Formatierte Ausgabe von MERKMALLISTE.
;
; ruft auf:             sich selbst
;
; aufgerufen von:       sich selbst
;                       ERGEBNIS-AUSGABE
;
; -MERKMALE-AUSGEBEN------------------------------------------------

        (COND ((NULL MERKMALLISTE)
               (TERPRI))
              ( T  (TERPRI)
                   (BLANKS 3)
                   (PRINC (CAR MERKMALLISTE))
                   (MERKMALE-AUSGEBEN
                          (CDR MERKMALLISTE))) ))

; -MERKMALE-AUSGEBEN------------------------------------------------




  (DEFUN PLAN-AUSGEBEN (PLAN)
; ---------------------------------------------------------------------
; Wert:                 'FERTIG
```

```
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLAN
;                       Auszugebender nichtlinearer Plan.
;
; Nebeneffekte:         Formatierte Ausgabe einer Kopie von PLAN (mit
;                       eigenem internen Speicherbereich), die
;                       mit PLAN bis auf die Namen der einzelnen Plan-
;                       knoten identisch ist. Die Knoten der Plan-Kopie
;                       tragen, dem Zeitpunkt ihres Einbaus in den
;                       Plan entsprechend, bei der Ausgabe folgende
;                       Namen: KNOTEN-START, KNOTEN-ZIEL, KNOTEN-1 (=
;                       der als erster in den Plan eingebauter Kno-
;                       ten), KNOTEN-2, KNOTEN-3 ...
;                       PLAN bleibt dabei in seiner urspruenglichen
;                       Form, auch was die Namen der einzelnen Knoten
;                       anbelangt, unveraendert.
;                       Die Knoten werden gemaess der bestehenden
;                       (partiellen) Operator-Ordnung ausgegeben. Das
;                       heisst allerdings nicht, dass der Startknoten
;                       immer als erster Knoten ausgegeben wird. Ist
;                       der Startknoten vom Zielknoten weniger ent-
;                       fernt als andere Planknoten, was durchaus der
;                       Fall sein kann, so werden diese anderen Plan-
;                       knoten vor dem Startknoten ausgegeben. Dies
;                       spielt aber nur eine sekundaere Rolle. Ent-
;                       scheidend ist die Operator-Ordnung gemaess
;                       den VORGAENGER-Eintraegen der einzelnen Kno-
;                       ten. Der Zielknoten bildet aber auf jeden
;                       Fall das Schlusslicht bei der Ausgabe.
;
; ruft auf:             PLAN-KOPIEREN
;                       BESTIMME-ZIELKNOTEN
;                       BESTIMME-DIREKTE-VORGAENGER
;                       ALTER
;                       PLAN-AUSGEBEN-HILF
;
; aufgerufen von:       ERGEBNIS-AUSGABE
;
; -PLAN-AUSGEBEN-----------------------------------------------------

      (LET* ((PLANKNOTENLISTE
                    (PLAN-KNOTENLISTE
                                (PLAN-KOPIEREN PLAN)))
             (ZIELKNOTEN
                    (BESTIMME-ZIELKNOTEN
                                PLANKNOTENLISTE))
             (GENERATION
                    (BESTIMME-DIREKTE-VORGAENGER
                                ZIELKNOTEN))
             (SORTIERTE_KNOTENLISTE
                    (SORT PLANKNOTENLISTE
                          #'(LAMBDA (K1 K2)
                                (< (ALTER K1)
                                   (ALTER K2))) )))
            (SETF (KNOTEN-NAME (CAR SORTIERTE_KNOTENLISTE))
                  'START)
            (SETF (KNOTEN-NAME ZIELKNOTEN)
                  'ZIEL)
            (DO ((RESTLISTE      (CDDR SORTIERTE_KNOTENLISTE)
                                 (CDR RESTLISTE))
```

```
                 (I             1
                                (+ I 1)))
               ((NULL RESTLISTE))
               (SETF (KNOTEN-NAME (CAR RESTLISTE))
                     I))
            (TERPRI)
            (PLAN-AUSGEBEN-HILF
                  GENERATION
                  (BESTIMME-DIREKTE-VORGAENGER
                          GENERATION)
                  (LIST ZIELKNOTEN))) )

; -PLAN-AUSGEBEN-----------------------------------------------------



  (DEFUN PLAN-AUSGEBEN-HILF (GENERATION
                             NAECHSTE_GENERATION
                             AUSGABELISTE)
; ------------------------------------------------------------------
; Wert:                 'FERTIG
;
; Globale Variablen:    --
;
; Eingangsparameter:      GENERATION
;                       Betrachtete Knotengeneration der auszugebenden
;                       Kopie des Loesungs-Plans.
;
;                         NAECHSTE_GENERATION
;                       Direkte Vorgaenger-Generation zu GENERATION in
;                       der auszugebenden Kopie des Loesungs-Plans.
;
;                         AUSGABELISTE
;                       Auszugebende Planknoten der Plan-Kopie des
;                       Loesungs-Plans. Die Ausgabe  der Knoten er-
;                       folgt in der Reihenfolge ihres Auftretens in
;                       AUSGABELISTE, welche in PLAN-AUSGEBEN-HILF
;                       als Nebeneffekt bestimmt wird.
;
; Nebeneffekte:         Formatierte Ausgabe saemtlicher Knoten der
;                       "umgetauften" Kopie des Loesungs-Plans.
;
; ruft auf:             sich selbst
;                       UNTERSTREICHEN
;                       KNOTEN-AUSGEBEN
;                       BESTIMME-DIREKTE-VORGAENGER
;
; aufgerufen von:       sich selbst
;                       PLAN-AUSGEBEN
;
; -PLAN-AUSGEBEN-HILF------------------------------------------------

        (COND ((AND
                   (NULL GENERATION)
                   (NULL NAECHSTE_GENERATION))
               (UNTERSTREICHEN 68)
               (KNOTEN-AUSGEBEN AUSGABELISTE)
               (UNTERSTREICHEN 68)
               'FERTIG)
              ((NULL GENERATION)
               (LET ((NICHTMARKIERTE_KNOTEN
```

```
                          (REMOVE-IF
                             #'(LAMBDA (PLANKNOTEN)
                                   (MEMBER PLANKNOTEN
                                           AUSGABELISTE))
                             NAECHSTE_GENERATION)))
                    (PLAN-AUSGEBEN-HILF
                            NICHTMARKIERTE_KNOTEN
                            (BESTIMME-DIREKTE-VORGAENGER
                                    NICHTMARKIERTE_KNOTEN)
                            AUSGABELISTE)))
              ( T  (PLAN-AUSGEBEN-HILF
                           (CDR GENERATION)
                           NAECHSTE_GENERATION
                           (CONS (CAR GENERATION)
                                 AUSGABELISTE))) ))

; -PLAN-AUSGEBEN-HILF-------------------------------------------------



  (DEFUN TEXT-AUSGABE1 ()
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:    --
;
; Nebeneffekte:         Ausgabe von erlaeuterndem Text.
;
; ruft auf:             BLANKS
;
; aufgerufen von:       FKNLP
;
; -TEXT-AUSGABE1-----------------------------------------------------

      (TERPRI)
      (PRINC '|    PLAN-GENERIERUNG FUER DAS GESTELLTE PROBLEM:|)
      (TERPRI)
      (BLANKS 4)
      (UNTERSTREICHEN 44)
      (TERPRI)
      (PRINC '|  Im folgenden wird fuer jeden aktuell betrachte-|)
      (TERPRI)
      (PRINC '|  ten (unvollstaendigen) nichtlinearen Plan ange-|)
      (TERPRI)
      (PRINC '|  zeigt:|)
      (TERPRI)
      (PRINC '|  - Nummer (fortlaufend; bei 0 beginnend),|)
      (TERPRI)
      (PRINC '|  - Knoten-Anzahl (mit Start- und Zielknoten),|)
      (TERPRI)
      (PRINC '|  - Anzahl der bereits generierten, aber noch nicht|)
      (TERPRI)
      (PRINC '|    untersuchten (unvollst.) nichtlinearen Plaene.|)
      (TERPRI)
      (PRINC '|  [bis zur eventuellen Ausgabe eines gesuchten op-|)
      (TERPRI)
      (PRINC '|   timalen Loesungs-Plans !]|)
      (TERPRI))
```

```
; -TEXT-AUSGABE1-----------------------------------------------------



  (DEFUN TEXT-AUSGABE2 (ZAEHLER
                        KNOTEN_ANZAHL
                        LAENGE_PLANWARTESCHLANGE)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      ZAEHLER
;                       Auszugebende aktuelle (fortlaufende) Nummer
;                       des aktuell betrachteten Plans.
;
;                         KNOTEN_ANZAHL
;                       Auszugebende Anzahl von Knoten des aktuell
;                       betrachteten Plans.
;
;                         LAENGE_PLANWARTESCHLANGE
;                       Auszugebende Anzahl der zum aktuellen Zeit-
;                       punkt in der Plan-Warteschlange verwalteten
;                       Plaene.
;
; Nebeneffekte:         Ausgabe der uebergebenen Argumente mit erlaeu-
;                       terndem Text.
;
; ruft auf:             --
;
; aufgerufen von:       PLANEN
;
; -TEXT-AUSGABE2-----------------------------------------------------

      (TERPRI)
      (PRINC '|Fortlaufende Nr.:   |)
      (PRINC ZAEHLER)
      (TERPRI)
      (PRINC '|Knoten-Anzahl:      |)
      (PRINC KNOTEN_ANZAHL)
      (TERPRI)
      (PRINC '|Plan-Warteschlange: |)
      (PRINC LAENGE_PLANWARTESCHLANGE)
      (TERPRI))

; -TEXT-AUSGABE2-----------------------------------------------------



  (DEFUN UNTERSTREICHEN (LAENGE)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      LAENGE
;                       Zahl der auszugebenden "-"
;
; Nebeneffekte:         Ausgabe von "-" (LAENGE mal) mit abschliessen-
;                       dem Zeilensprung.
```

```
;
; ruft auf:             --
;
; aufgerufen von:       ERGEBNIS-AUSGABE
;                       KNOTEN-AUSGEBEN
;                       PLAN-AUSGEBEN-HILF
;                       TEXT-AUSGABE1
;
; -UNTERSTREICHEN--------------------------------------------------

        (DO ((I     LAENGE     (- I 1)))
            ((ZEROP I) (TERPRI))
            (PRINC '|-|)))

; -UNTERSTREICHEN--------------------------------------------------



  (DEFUN VARIABLENBINDUNGEN-AUSGEBEN (PLANKNOTEN
                                      VARIABLENLISTE)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      PLANKNOTEN
;                       Bezugs-Planknoten zu VARIABLENLISTE.
;
;                         VARIABLENLISTE
;                       Liste (saemtlicher) auszugebender Variablen
;                       der Operator-Instanz PLANKNOTEN.
;
; Nebeneffekte:         Formatierte Ausgabe saemtlicher Variablen aus
;                       VARIABLENLISTE samt den per Wertconstraints
;                       direkt verbundenen Partner-Argumenten.
;
; ruft auf:             sich selbst
;                       BESTIMME-ACOD-HUELLE
;                       KONSTANTEN-AUSGEBEN
;                       KAPAARE-ORDNEN
;                       BESTIMME-DIREKTE-ABINDUNGEN
;                       BLANKS
;                       KAPAARE-AUSGEBEN
;
; aufgerufen von:       sich selbst
;                       KNOTEN-AUSGEBEN
;
; -VARIABLENBINDUNGEN-AUSGEBEN-------------------------------------

        (COND
           ((NULL VARIABLENLISTE))
           ( T  (LET*
                  ((VARIABLE
                      (CAR VARIABLENLISTE))
                   (BLANK_ANZAHL
                      (+ (LENGTH (SYMBOL-NAME VARIABLE))
                         24))
                   (KK_KV_CODS
                      (KAPAARE-ORDNEN
                         (BESTIMME-ACOD-HUELLE
                            (LIST
```

```
                (LIST PLANKNOTEN
                      VARIABLE))) ))
 (CODESIGNIERTE_KONSTANTEN
    (CAR KK_KV_CODS))
 (CODESIGNIERTE_VARIABLEN
    (CDADR KK_KV_CODS)))
(TERPRI)
(BLANKS 20)
(PRINC VARIABLE)
(COND
   ((NOT (NULL CODESIGNIERTE_KONSTANTEN))
    (PRINC '| =  |)
    (KONSTANTEN-AUSGEBEN
        (CDAR CODESIGNIERTE_KONSTANTEN)))
   ( T
    (LET*
      ((KK_KV_NONCODS
          (KAPAARE-ORDNEN
             (BESTIMME-ACOD-HUELLE
                (BESTIMME-DIREKTE-ABINDUNGEN
                   (CADR KK_KV_CODS)
                   'KNOTEN-NONCODESIGNATIONS))) )
       (NONCODESIGNIERTE_KONSTANTEN
          (CAR KK_KV_NONCODS))
       (NONCODESIGNIERTE_VARIABLEN
          (CADR KK_KV_NONCODS))
       (NONCODESIGNIERTE_ARGUMENTE
          (APPEND
             NONCODESIGNIERTE_KONSTANTEN
             NONCODESIGNIERTE_VARIABLEN)))
      (COND
         ((NOT (NULL CODESIGNIERTE_VARIABLEN))
          (PRINC '| =  |)
          (KAPAARE-AUSGEBEN
             CODESIGNIERTE_VARIABLEN
             BLANK_ANZAHL)))
      (COND
         ((NOT (NULL NONCODESIGNIERTE_ARGUMENTE))
          (COND
             ((NOT (NULL CODESIGNIERTE_VARIABLEN))
              (TERPRI)
              (BLANKS
                 (- BLANK_ANZAHL
                    4))) )
          (PRINC '| <> |)
          (KONSTANTEN-AUSGEBEN
             (MEHRFACHE-RAUS
               (MAPCAR
                  #'(LAMBDA (KKPAAR)
                       (CADR KKPAAR))
                  NONCODESIGNIERTE_KONSTANTEN))
             BLANK_ANZAHL)
          (COND
             ((NOT (NULL NONCODESIGNIERTE_KONSTANTEN))
              (TERPRI)
              (BLANKS
                   BLANK_ANZAHL)))
          (KAPAARE-AUSGEBEN
             NONCODESIGNIERTE_VARIABLEN
             BLANK_ANZAHL))) )))
(VARIABLENBINDUNGEN-AUSGEBEN
     PLANKNOTEN
```

```
                        (CDR VARIABLENLISTE))) )))

; -VARIABLENBINDUNGEN-AUSGEBEN-------------------------------------



  (DEFUN VORGAENGER-AUSGEBEN (KNOTENLISTE)
; ---------------------------------------------------------------------
; Wert:                 irrelevant!
;
; Globale Variablen:    --
;
; Eingangsparameter:      KNOTENLISTE
;                       Liste von Knoten, die den gesamten VORGAENGER-
;                       Eintrag eines Planknotens darstellt. Es sind
;                       nur die Namen der Knoten auszugeben.
;
; Nebeneffekte:         Formatierte Ausgabe der Namen saemtlicher Kno-
;                       ten aus KNOTENLISTE.
;
; ruft auf:             sich selbst
;                       BLANKS
;
; aufgerufen von:       sich selbst
;                       KNOTEN-AUSGEBEN
;
; -VORGAENGER-AUSGEBEN-----------------------------------------------

        (COND ((NULL KNOTENLISTE))
              ( T  (LET ((PLANKNOTEN
                                (CAR KNOTENLISTE))
                         (REST
                                (CDR KNOTENLISTE)))
                        (PRINC 'KNOTEN-)
                        (PRINC (KNOTEN-NAME
                                    PLANKNOTEN))
                        (COND ((NOT (NULL REST))
                               (TERPRI)
                               (BLANKS 20)
                               (VORGAENGER-AUSGEBEN
                                     (CDR KNOTENLISTE))) ))))

; -VORGAENGER-AUSGEBEN-----------------------------------------------



; =FUNKTIONEN ZUR PLAN-AUSGABE=======================================



; =FUNKTIONEN=========================================================
```

```
; ======================================================================
; =============   BEISPIEL FUER EINE PROBLEMSTELLUNG   ================
; ======================================================================

; Stand: 29.05.1990


; ACHTUNG:
; --------
; Hier ist in den unten gekennzeichneten Bereichen (zwischen '*'-Zeilen)
; fuer das jeweilige Planungsproblem eine entsprechende Modifikation
; durch den Benutzer durchzufuehren!!


; ======================================================================
;                          Problemspezifikation
;                          --------------------
; Beschreibung der gegebenen Startsituation und der erwuenschten Ziel-
; situation dadurch, dass die jeweils geltenden Merkmale aufgelistet
; werden!
; ======================================================================


  (DEFUN STARTOPERATOR ()
        (MAKE-OPERATOR
             :NACHBEDINGUNGEN
                       '(
; ----------------------------------------------------------------------
; Im folgenden sind die Merkmale der gegebenen STARTSITUATION anzugeben:
; **********************************************************************
; *                                                                    *
        (steht_auf C A)
        (steht_auf A TISCH)
        (steht_auf B TISCH)
        (ist_frei C)
        (ist_frei B)
; *                                                                    *
; **********************************************************************
                        )
               :LOKALE_CONSTRAINTS
                        '(
; ----------------------------------------------------------------------
; Im folgenden sind die lokalen Wert-Constraints, die in der
; STARTSITUATION Gueltigkeit haben, anzugeben:
; **********************************************************************
; *                                                                    *

; *                                                                    *
; **********************************************************************
                        )))



  (DEFUN ZIELOPERATOR ()
      (MAKE-OPERATOR
             :VORBEDINGUNGEN
                       '(
; ----------------------------------------------------------------------
; Im folgenden sind die Merkmale der erwuenschten FINALSITUATION anzu-
; geben:
```

```
; ************************************************************************
; *                                                                      *
       (steht_auf A B)
       (steht_auf B C)
; *                                                                      *
; ************************************************************************
                     )
            :LOKALE_CONSTRAINTS
                     '(
; ------------------------------------------------------------------------
; Im folgenden sind die lokalen Wert-Constraints, die in der
; FINALSITUATION Gueltigkeit haben, anzugeben:
; ************************************************************************
; *                                                                      *

; *                                                                      *
; ************************************************************************
                     )))



; ========================================================================
;                       Operatorspezifikation
;                       ---------------------
;    Beschreibung der einzelnen dem Planer zur Verfuegung stehenden
;    Planungsoperatoren (die Reihenfolge spielt keine Rolle)!
; ========================================================================


  (DEFUN OPERATOREN ()
      '(
; ------------------------------------------------------------------------
; An dieser Stelle ist die OPERATORBEZEICHNUNG des ersten Operators
; anzugeben:
; ************************************************************************
; *                                                                      *
           PUTON
; *                                                                      *
; ************************************************************************
                (MAKE-OPERATOR :VORBEDINGUNGEN
                               '(
; ------------------------------------------------------------------------
; Im folgenden sind die VORBEDINGUNGEN des ersten Operators anzu-
; geben:
; ************************************************************************
; *                                                                      *
     (steht_auf ?X ?Z)
     (ist_frei ?X)
     (ist_frei ?Y)
; *                                                                      *
; ************************************************************************
                               )
                                :NACHBEDINGUNGEN
                               '(
; ------------------------------------------------------------------------
; Im folgenden sind die NACHBEDINGUNGEN des ersten Operators anzu-
; geben:
; ************************************************************************
; *                                                                      *
     (steht_auf ?X ?Y)
     (ist_frei ?Z)
```

```
      (NICHT (steht_auf ?X ?Z))
      (NICHT (ist_frei ?Y))
; *                                                                    *
; **********************************************************************
                              )
                               :LOKALE_CONSTRAINTS
                              '(
; ----------------------------------------------------------------------
; Im folgenden sind die Non-Codesignation-CONSTRAINTS (Symbol: '<>')
; des ersten Operators anzugeben:
; **********************************************************************
; *                                                                    *
      (<> ?X ?Y)
      (<> ?X ?Z)
      (<> ?Y ?Z)
      (<> ?X TISCH)
; *                                                                    *
; **********************************************************************
                              ))

; ----------------------------------------------------------------------
; An dieser Stelle ist die OPERATORBEZEICHNUNG des zweiten Operators
; anzugeben:
; **********************************************************************
; *                                                                    *
           NEWTOWER
; *                                                                    *
; **********************************************************************
                 (MAKE-OPERATOR :VORBEDINGUNGEN
                              '(
; ----------------------------------------------------------------------
; Im folgenden sind die VORBEDINGUNGEN des zweiten Operators anzu-
; geben:
; **********************************************************************
; *                                                                    *
      (steht_auf ?X ?Z)
      (ist_frei ?X)
; *                                                                    *
; **********************************************************************
                              )
                               :NACHBEDINGUNGEN
                              '(
; ----------------------------------------------------------------------
; Im folgenden sind die NACHBEDINGUNGEN des zweiten Operators anzu-
; geben:
; **********************************************************************
; *                                                                    *
      (steht_auf ?X TISCH)
      (ist_frei ?Z)
      (NICHT (steht_auf ?X ?Z))
; *                                                                    *
; **********************************************************************
                              )
                               : LOKALE_CONSTRAINTS
                              '(
; ----------------------------------------------------------------------
; Im folgenden sind die Non-Codesignation-CONSTRAINTS (Symbol: '<>')
; des zweiten Operators anzugeben:
; **********************************************************************
; *                                                                    *
      (<> ?X ?Z)
      (<> ?X TISCH)
```

```
      (<> ?Z TISCH)
; *                                                                    *
; **********************************************************************
                                ))
))
```