PROJEKTARBEIT

Aufbau einer domänenunabhängigen Fallbasis und Fallauswahl mit Zielabhängigkeiten in CAPLAN/CBC

Jochem Hüllen

Betreuung
Prof. Dr. Michael M. Richter
Doktorand Héctor Muñoz A.

Universität Kaiserslautern
Fachbereich Informatik

Januar 1995

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

CAPLAN [4] ist ein SNLP-basiertes Planungssystem. Der Suchraum für ein Planungsproblem ist exponentiell, da es für jedes Ziel eine Menge von alternativen Schritten gibt und für jeden Konflikt mehrere Alternativen, die den Konflikt lösen können [6]. Außerdem ist die Suchtiefe nicht bekannt, da die Anzahl der nötigen Schritte zur Lösung eines Problems in der Regel nicht vorhergesagt werden kann. Ein möglicher Ansatz zur Steuerung des Suchverfahrens ist das Benutzen von bereits bekannten Fällen. Diese enthalten Entscheidungen, die getroffen wurden, um das zugehörige Problem zu lösen. Diese Entscheidungen werden auf ein neues Problem übertragen, um dieses zu lösen. Eine schwierige Aufgabe dieses Ansatzes ist die Bestimmung von geeigneten Fällen. In dieser Arbeit wird eine Architektur einer Fallbasis und ein Verfahren für eine effiziente Fallauswahl vorgestellt.

In [2] wurde eine hybride fallbasierte Komponente vorgestellt, die den Fertigungsprozeß von mechanischen Werkstücken steuert. Eine Hierarchie zur Klassifizierung der Werkstücke wurde von Experten dieser Domäne definiert, sie dient als Index der Fallbasis. Leider stehen solche Informationen nicht in jeder Domäne zu Verfügung, oder es kann sehr aufwendig sein, diese Information herauszufinden. Außerdem spiegeln die Kriterien solcher Hierarchien in vielen Domänen die Übertragbarkeit der Fälle nur unbefriedigend wieder. Hier wird eine Architektur vorgestellt, die domänen-unabhängig ist. Eine mehrstufige Baumstruktur dient als Index auf die Fälle der Fallbasis.

Experimente haben gezeigt [5], daß diese Architektur der Fallbasis für die Arbeitsplanungsdomäne geeigneter ist als die, die in [3] vorgestellt wurde.

In dieser Arbeit entstammen alle Beispiele der relativ einfachen Domäne Klötzchen-Welt (Blocksworld), um die Konzepte leicht verständlich zu machen. Die domänenunabhängige Struktur, die hier vorgestellt wird, eignet sich auch für komplexere Domänen, wie z.B. die Arbeitsplanungsdomäne. Der Vorteil dieser Architektur und damit in erster Linie die Effizienzerhöhung steigt sogar mit der Komplexität der Domäne, bzw. mit der Komplexität der betrachteten Probleme.

Kapitel 2

Allgemeines zur Fallbasis in CAPLAN/CBC

2.1 Funktion der Fallbasis

Ziel des fallbasierten Ansatzes ist es, ein neues Problem auf alte, ähnliche Probleme, deren Lösungen bereits bekannt sind, zu reduzieren. Das heißt, es werden möglichst viele Informationen in Form von Entscheidungen der alten Fälle auf das neue Problem übertragen, um dieses effizient zu lösen. Die Einbindung der fallbasierten Komponente und der Ablauf bei der Lösung eines Problems ist in Abb. 2.1 dargestellt.

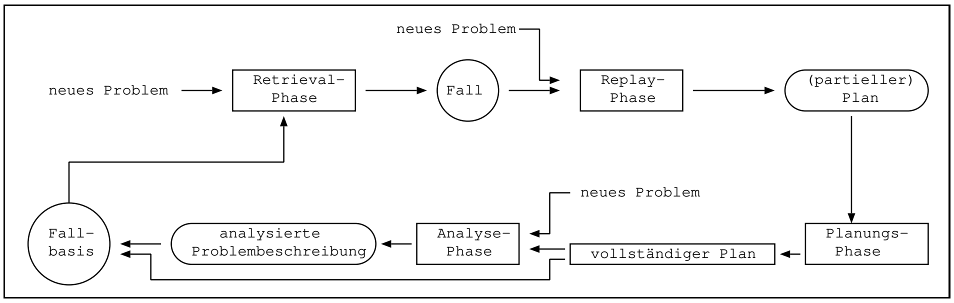

Abbildung 2.1: Prozeßablauf von CAPLAN/CBC

Ein neues Problem wird gegeben und ähnliche Fälle werden aus der Fallbasis ausgewählt. In der Replay-Phase werden die Entscheidungen der ausgewählten Fälle, die konsistent mit dem neuen Problem sind, auf dieses übertragen. In der Planungs-Phase wird der in der Replay-Phase erzeugte Plan vervollständigt. Dann wird das neue Problem, zusammen mit der Lösung analysiert. Der erzeugte Plan wird dann zusammen mit der analysierten Problembeschreibung als ein Fall gespeichert. Die analysierte Problembeschreibung dient als Kriterium für das Speichern in der Fallbasis als Fall (vgl. Kap. 4.) und für eine mögliche Auswahl in der Retrieval-Phase (vgl. Kap. 5.).

2.2 Effizienzproblem der Fallauswahl

Anforderung an die Fallauswahl: Es soll möglichst schnell eine Anzahl von Fällen in der Fallbasis gefunden werden, die es zusammen ermöglichen, daß in der Replay-Phase ein möglichsts großer Teil des neuen Problems gelöst werden kann. Dazu müssen in der Fallbasis relevante Fälle vorhanden sein, und diese müssen möglichst effizient ausgewählt werden.
Ein neues Problem kann aus mehreren Teilproblemen bestehen, die jeweils von unterschiedlichen Fällen oder auch nur von Teilen dieser Fälle abgedeckt werden. Daraus ergibt sich ein kombinatorisches Problem, da diese Möglichkeiten getestet werden müssen, bis eine zufriedenstellende gefunden wurde oder bis alle getestet wurden. Zusätzlich steigt mit der Komplexität der Domäne und mit der Komplexität der betrachteten Probleme die Anzahl der nötigen Fälle in der Fallbasis, um für potentielle Probleme relevante Fälle bereitzuhalten. Somit hat eine triviale (z.B. einfache sequentielle) Suche nach ähnlichen Fällen exponentiellen Aufwand und damit in der Regel kein befriedigendes Laufzeitverhalten.

2.3 Lösung durch Index auf Fälle

In CAPLAN/CBC wird das Effizienzproblem gelöst, indem die Fallbasis durch einen mehrstufigen Baum so strukturiert wird, daß bei der Suche möglichst schnell möglichst viele Fälle von der Betrachtung ausgeschlossen werden, indem garantiert wird, daß sie nicht in Frage kommen können. Um das Konzept des Suchbaumes zu verstehen, muß zuerst das Ähnlichkeitskriterium für Probleme, welches in CAPLAN/CBC benutzt wird, erläutert werden.

2.3.1 Ähnlichkeitskriterium für Problemen

Ein Planungsproblem in einer bestimmten Domäne ist durch seinen Startzustand und durch seinen Zielzustand charakterisiert. Diese beiden Zustände werden durch eine Anzahl von Prädikaten beschrieben. Nun liegt es nahe zu definieren, daß zwei Fälle um so ähnlicher sind, je mehr gemeinsame Prädikate sie in den beiden Zuständen haben. Bei der Fallauswahl würde man also den Fall nehmen, der die größte Anzahl von Prädikaten mit dem neuen Problem gemeinsam hat. Nun kann es aber vorkommen, daß bei einem Fall einige Prädikate des Startzustandes irrelevant für das Problem sind, und eine Überlappung bei diesen wäre demnach auch irrelevant. Also versucht man nur die Aussagen über einen Fall zu benutzen, die ihn charakterisieren, dies nennt man in Anlehnung an [3] Footprinting.

2.3.2 Footprinting und Zielabhängigkeiten

Es gibt oft Probleme, die sich in völlig unabhängige Teile zerlegen lassen. Das heißt, daß diese Teile unabhängig voneinander gelöst werden können und die Teillösungen einfach zu der Gesamtlösung zusammengenommen werden. Es macht Sinn, diese Teile einzeln zu betrachten, da z.B. ein neues Problem aus einzelnen Teilen verschiedener Fälle zusammengesetzt sein kann und somit bei diesem Problem nur diese Teillösungen der Fälle interessant sind. Aus der Problembeschreibung und der Lösung, also einem Plan, läßt sich in der Analysephase der Footprint eines Falles bestimmen, der die Irrelevanz von Prädikaten und die Unabhängikeit von Teilen berücksichtigt.
Die Menge der Einzelziele des Zielzustandes wird in unabhängige Teilmengen zerlegt, das heißt, daß die Ziele in einer Teilmenge interagieren, aber Ziele aus verschiedenen Teilmengen unabhängig sind. Eine solche Teilmenge heißt zusammenhängende Komponente. Für jede dieser Teilmengen werden die Prädikate des Startzustandes bestimmt, die für die Ziele dieser Teilmenge relevant sind, also diejenigen, die im Plan zur Lösung benutzt wurden.
In CAPlan/CbC [5] werden zusätzlich noch Abhängikeiten zwischen den interagierenden Zielen betrachtet. Die einzelnen Ziele in einer Teilmenge werden (partiell) geordnet. Wenn also bei dem zugehörigen Plan eine bestimmte Reihenfolge der Erfüllung der Ziele notwendig ist, wird diese in die Ordnung übernommen.
Bei der Strukturierung der Fallbasis werden nun diese Informationen des Footprints und der Zielabhängigkeiten eines Falles ausgenutzt (vgl. Kapitel 3.). Der Aufbau, des in den folgenden Kapiteln beschriebenen Indexes auf die Fälle, ermöglicht eine effiziente Fallauswahl, indem die Fälle bezüglich des Footprints und der Zielabhängigkeiten so einsortiert werden, daß bei der Suche nach ähnlichen Fällen möglichst schnell möglichst viele Fälle ausgeschlossen werden können.
Die Konzepte von Lösung, Footprint und Zielabhängigkeiten eines Problems werden im folgenden Beispiel verdeutlicht.
Alle Beispiele entstammen der Domäne Klötzchen-Welt (Blocksworld) mit den Prädikaten $On(Argument, Argument)$ und $Clear(Argument)$.

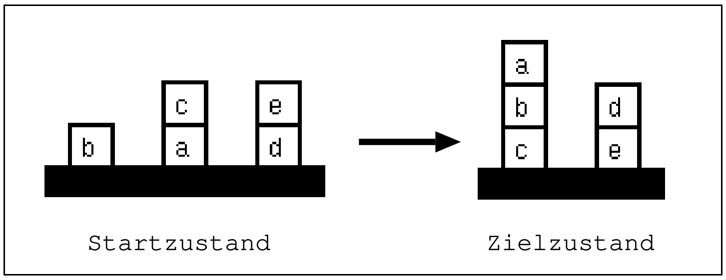

Abbildung 2.2: Problembeschreibung

In Abb. 2.2 ist das Problem bildlich dargestellt. Die Beschreibung durch Prädikate ist für den Startzustand {*On(a,t), On(b,t), On(c,a), On(d,t), On(e,d), Clear(b), Clear(c), Clear(e)* } und für den Zielzustand {*On(a,b), On(b,c), On(c,t), On(d,e), On(e,t), Clear(a), Clear (d)* }. In Abb. 2.3 ist ein möglicher Plan als Lösung dargestellt. Daraus resultiert folgender Footprint:
Es gibt zwei zusammenhängende Komponenten *K1* und *K2*. *K1* enthält die Ziele {*On(d,e), On(e,t), Clear(d)* }, *K2* enthält die übrigen. Die für *K1* relevanten Prädikate des Startzustandes sind {*On(d,t), On(e,d), Clear(e)* }, alle anderen sind für *K2* relevant. Den Zielen von *K1* entsprechen im Plan die Schritte von Zweig 1 und den Zielen von *K2* die Schritte in Zweig 2. Außerdem werden folgende Zielabhängigkeiten analysiert: $On(c,t)-> On(b,c)-> On(a,b) \quad , \quad On(e,t)-> On(d,e)$

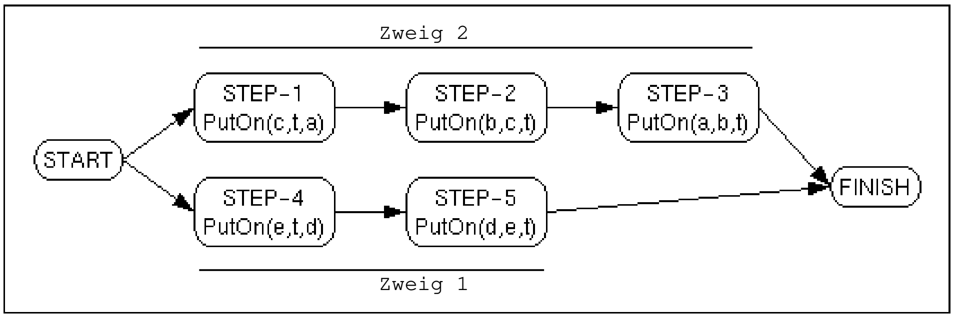

Abbildung 2.3: Plan mit Abhängigkeiten zwischen den einzelnen Schritten

Kapitel 3

Aufbau der Fallbasis in CAPLAN/CBC

3.1 Allgemeines zum Aufbau

Die zusammenhängenden Komponenten eines Falles werden einzeln in den Suchbaum einsortiert, und an den Blättern wird auf den zugehörigen Fall verwiesen. Der Index besteht aus fünf Stufen (vergl. Abb. 3.1), in denen die zusammenhängenden Komponenten nach folgenden Kriterien unterschieden werden:

1. Anzahl der interagierenden Ziele
2. Klassenparametrisierte Form
3. Parametrisierte Form
4. Ordnung der Ziele
5. Startzustand

3.1.1 Klassenparametrisierte Form

Bei der klassenparametrisierten Form eines Prädikates werden die Argumente durch ihren Typ ersetzt. Das hat zur Folge, daß zwei Prädikate, die in der klassenparametrisierten Form nicht übereinstimmen, auf keinen Fall unifiziert werden können. Der Test, ob zwei Mengen von Prädikaten unifiziert werden können, ist wesentlich komplexer, als der Test, ob zwei Mengen von klassenparametrisierten Prädikaten gleich sind (siehe 4.1 Unterpunkte 2., 3.). Die zweite Stufe soll also eine effiziente Vorauswahl aller Fälle realisieren, die die gleiche Anzahl von Zielen haben.

Beispiele : Seien a, b vom Typ *Block* und t vom Typ *Table*.

Prädikat	klassenparametrisierte Form
$On(a,t)$	$On(Block, Table)$
$Clear(b)$	$Clear(Block)$

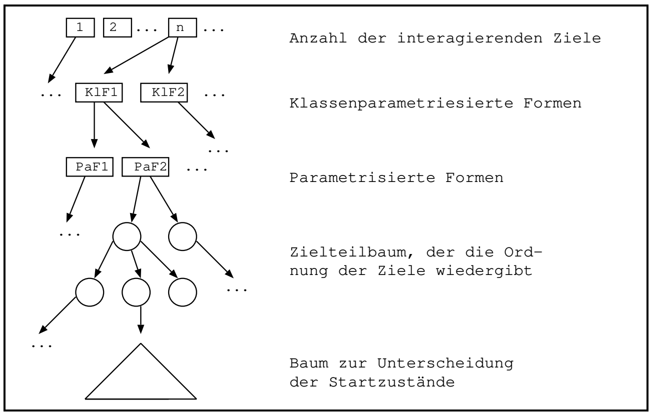

Abbildung 3.1: Abstrakter Aufbau der Stuktur

3.1.2 Parametrisierte Form

Bei der parametrisierten Form eines Prädikates werden die Argumente durch Variablen ersetzt.

Beispiele : Seien a, b, c, d, i, j, k, l, x, y, z Konstanten vom Typ *Block* und seien V_a ... V_z Variablen.

Nr.	Prädikat	parametrisierte Form
1	$On(a,b)$	$On(V_a, V_b)$
2	$On(c,d)$	$On(V_c, V_d)$
3	$On(i,j)$	$On(V_i, V_j)$
4	$On(k,l)$	$On(V_k, V_l)$
5	$On(x,y)$	$On(V_x, V_y)$
6	$On(y,z)$	$On(V_y, V_z)$

Werden nun die parametrisierten Formen verglichen, ergibt sich: $\{1,2\}$ und $\{3,4\}$ sind unifizierbar mit den Bindungen $\{(V_a, V_i), (V_b, V_j), (V_c, V_k), (V_d, V_l)\}$, während $\{1,2\}$ und $\{5,6\}$ nicht unifizierbar sind, da die Bindungen (V_b, V_y) und (V_c, V_y) nötig wären

und somit V_b und V_c an das gleiche Objekt gebunden würde. **Dabei ist zu beachten, daß im Gegensatz zur üblichen Unifikation hier verschiedene Variablen an verschiedene Objekte gebunden werden müssen.** Diese zusätzliche Regel spiegelt den Sachverhalt wieder, daß bei Problembeschreibungen unterschiedliche Objektnamen in der Regel unterschiedliche Objekte beschreiben.
$\{1, 2\}$ und $\{3, 4\}$ kämen also in einen Ast des Baumes, während $\{5, 6\}$ bezüglich der klassenparametrisierten Form zwar gleich ist, aber in der parametrisierten Form unterschiedlich. Im folgenden werden die Konstanten durch Variablen mit dem gleichen Namen ersetzt.

3.2 Beschreibung der Struktur

Um den Aufbau des Indexes zu verstehen, wird nun eine Fallbasis mit sieben einfachen Fallbeispielen betrachtet, die jeweils aus nur einer zusammenhängenden Komponente bestehen.

Name	Startzustand	Zielzustand	Zielabhängikeiten
$Frei$	$On(a,b)$, $Clear(a)$	1. $Clear(b)$	-
$Turm\_2$	$On(a,t)$, $Clear(a)$, $On(b,t)$, $Clear(b)$	1. $On(a,b)$	-
$Turm\_3\_A$	$On(a,t)$, $Clear(a)$, $On(c,b)$, $Clear(c)$, $On(b,t)$	1. $On(a,b)$, 2. $On(b,c)$, 3. $On(c,t)$	3.− >2.− >1.
$Turm\_3\_B$	$On(z,y)$, $Clear(z)$, $On(y,x)$, $On(x,t)$	1. $On(x,y)$, 2. $On(y,z)$, 3. $On(z,t)$	3.− >2.− >1.
2_$Tuerme_A$	$On(a,t)$, $Clear(a)$, $On(d,c)$, $Clear(d)$, $On(c,b)$, $On(b,t)$	1. $On(a,b)$, 2. $On(c,d)$, 3. $On(d,t)$	3.− >2.− >1.
2_$Tuerme_B$	$On(b,t)$, $Clear(b)$, $On(a,d)$, $Clear(a)$, $On(d,c)$, $On(c,t)$	1. $On(a,b)$, 2. $On(c,d)$, 3. $On(d,t)$,	1.− >3.− >2.
2_$Tuerme_C$	$On(a,t)$, $Clear(a)$, $On(c,t)$, $Clear(c)$, $On(d,b)$, $Clear(b)$, $On(b,t)$	1. $On(a,b)$, 2. $On(c,d)$, 3. $On(d,t)$	3.− >2. 3.− >1.

Dabei sind die Zielabhängigkeiten durch die Analyse der zugehörigen Pläne entstanden. Beim Beispiel *Turm_3_A* wurde also das Ziel *On(c,t)* vor dem Ziel *On(b,c)* erzeugt.
Betrachtet man zunächst nur die beiden Fälle *Turm_2* und *Turm_3_A*, dargestellt in Abb. 3.2 (a), so unterscheiden sie sich in der Anzahl der Ziele und liegen demnach in verschieden Ästen der 1. Stufe.

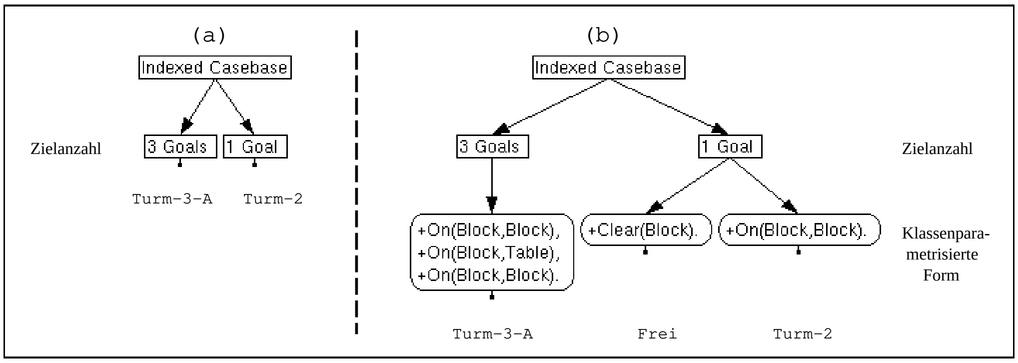

Abbildung 3.2: (a) Unterscheidung bezüglich Anzahl der Ziele (b) Unterscheidung bezüglich der klassenparametrisierten Form

Nimmt man den Fall *Frei* hinzu, der auch ein Ziel hat, so unterscheidet dieses sich jedoch bezüglich der klassenparametrisierten Form zu *Turm_2*. Diese beiden Fälle werden demnach in der 2. Stufe unterschieden, siehe Abb. 3.2 (b).
Nun betrachtet man dazu den Fall *2_Tuerme_A*. Er stimmt in den ersten beiden Stufen mit *Turm_3_A* überein, hat aber eine andere parametrisierte Form, siehe Abb. 3.3.

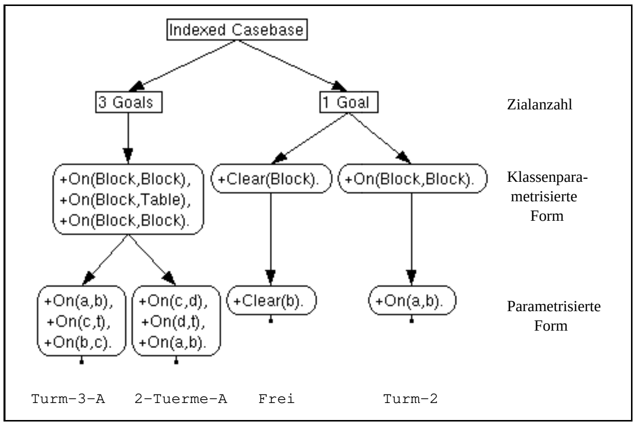

Abbildung 3.3: Unterscheidung bezüglich der parametrisierten Form

Nimmt man noch die Fälle 2_*Tuerme_B* und 2_*Tuerme_C* hinzu, so stimmen alle drei Fälle mit 2 Tuermen in den ersten 3 Stufen überein. Sie unterscheiden sich jedoch in der Reihenfolge der Ziele, siehe Abb. 3.4 (a). Der Zielteilbaum wird wie folgt aufgebaut: Verfolgt man einen Ast von der Wurzel zu einem Blatt und sammelt die Prädikate in den Knoten des Astes auf, so erhält man die gesamte Menge der Ziele der Fälle, auf die von diesem Blatt aus verwiesen wird. **Dabei entspricht der Vater-Sohn-Beziehung die (partielle) Ordnung der Ziele.** Das bedeutet, daß die Ziele der Söhne erst erfüllt werden können, wenn die Ziele des Vaters bereits erfüllt sind.

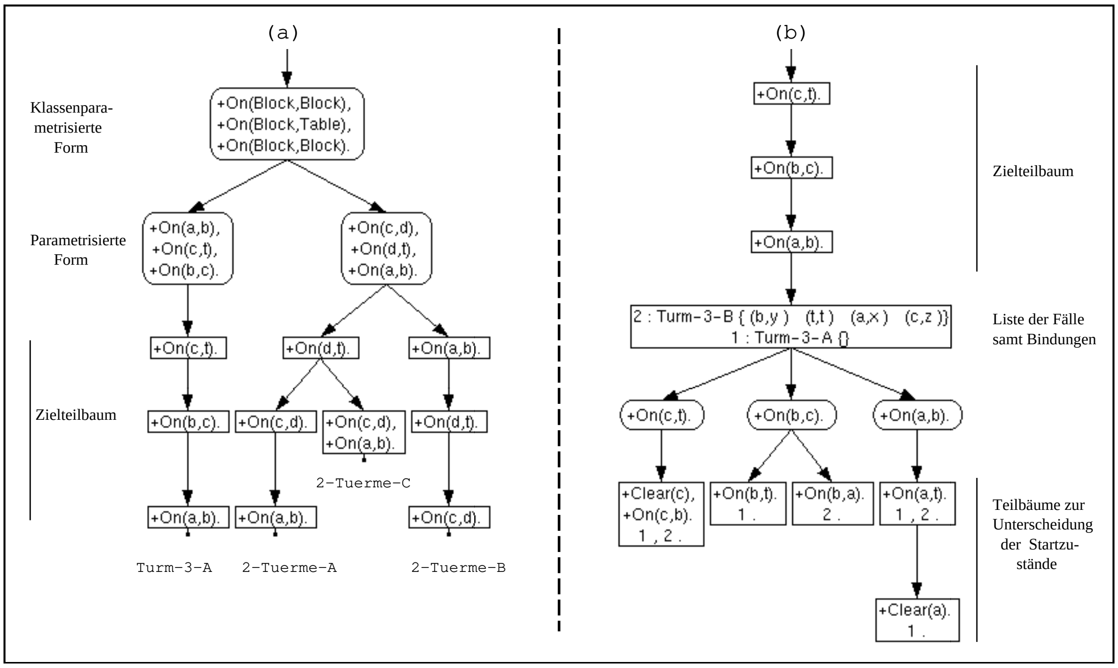

Abbildung 3.4: (a) Unterscheidung bezüglich der Zielreihenfolge (b) Unterscheidung bezüglich des Startzustandes

Betrachtet man zusätzlich den letzten Fall *Turm_3_B*, so stimmt er in den ersten vier Stufen mit Fall *Turm_3_A* überein, wobei die folgenden Variablenbindungen entstehen: $\{(a, x), (b, y), (c, z)\}$. Die beiden Fälle werden in der 5. Stufe unterschieden, siehe Abb. 3.4 (b). Die Bäume zur Unterscheidung des Startzustandes werden wie folgt aufgebaut: Für jeden einzelnen Knoten im Zielteilbaum wird ein Baum erstellt, der die für die Ziele in dem jeweiligen Knoten relevanten Prädikate enthält. Dabei werden möglichst viele Gemeinsamkeiten der Fälle, die in diesem Teilbaum enthalten sind, berücksichtigt. In den einzelnen Knoten ist vermerkt, welche der Fälle diese Prädikate im Startzustand enthalten (hier durch die Nummer des Falles).

3.3 Eigenschaften der Struktur

Durch die zahlreichen und teilweise komplexen Unterscheidungskriterien wird garantiert, daß, auch bei einer großen Anzahl von Fällen in einer Fallbasis, diese ausreichend separiert werden. Durch den Stufenaufbau ist es möglich, zuerst mit sehr einfachen Vergleichen eine große Anzahl von Fällen von der Suche auszuschließen und nach und nach durch komplexere Vergleichsmethoden die nun stark reduzierte Anzahl von Fällen genauer zu untersuchen. Dadurch ist eine sehr effiziente Fallauswahl möglich. Die Operationen auf dieser Struktur (Einfügen eines Falles und die Fallauswahl) werden in den folgenden Kapiteln erläutert.

Kapitel 4

Einfügen eines Falles in CAPLAN/CBC

4.1 Beschreibung des Algorithmus

Zuerst wird der Footprint und die Abhängigkeiten der Ziele des neuen Problems bestimmt. Dadurch entstehen zusammenhängende Komponenten. Diese werden nun einzeln in die Fallbasis eingefügt. Dies geschieht gemäß des Stufenaufbaus des Index (vgl. 3.1) in fünf Schritten :

1. Für die Einsortierung in der 1. Stufe muß nur die Anzahl der interagierenden Ziele betrachtet werden. Existieren bereits Fälle mit dieser Anzahl, so wird in diesem Ast fortgefahren. Wenn nicht, wird ein neuer Knoten erzeugt und dort fortgefahren.

2. Nun wird die klassenparametrisierte Form der Ziele gebildet. Diese wird sukzessive mit den bereits in diesem Teilbaum vorhanden klassenparametrisierten Formen auf Gleichheit getestet. Bei einem positiven Vergleich wird in diesem Ast bezüglich der nächsten Stufe einsortiert. War kein Vergleich positiv, so wird ein neuer Knoten eingefügt und dort fortgefahren.

3. Jetzt wird die parametrisierte Form gebildet und im Prinzip wie in der vorhergehenden Stufe verfahren. Anstatt auf Gleichheit zu testen wird hier auf Unifizierbarkeit getestet [1], da die Variablen gebunden werden müssen und dabei Backtracking notwendig sein kann. Dazu ein Beispiel :
Parametrisierte Form eines bereits in der Fallbasis vorhanden Falles :
$\{On(a,b), On(b,c)\}$
Parametrisierte Form des neu einzufügenden Falles :
$\{On(x,y), On(z,x)\}$

[1] Wie schon in 3.1.2 erwähnt, wird hier bei der Unifikation eine zusätzliche Bedingung verlangt: Unterschiedliche Variablen müssen an unterschiedliche Objekte gebunden werden.

Wird nun zuerst versucht $On(a, b)$ und $On(x, y)$ zu unifizieren, so entstehen die Bindungen $\{(a, x), (b, y)\}$, und dann können $On(b, c)$ und $On(z, x)$ nicht mehr unifiziert werden, da x auch noch an c gebunden werden müßte. Allerdings sind die beiden Mengen mit den Bindungen $\{(a, z), (b, x), (c, y)\}$ unifizierbar.
Die in dieser Stufe entstandenen Bindungen werden allerdings wieder verworfen, da in der nächsten Stufe unter Berücksichtigung der Ordnung andere entstehen können. Hier wird also nur festgestellt, ob die Unifikation möglich ist.

4. Nun wird die einzufügende Komponente sukzessive in den Zielteilbaum einsortiert. Dazu wird in jedem Schritt die Menge Mz der Ziele bestimmt, deren Vorgänger bezüglich der durch die Abhängigkeit der Ziele festgelegten Ordnung bereits in den Vaterknoten im Zielteilbaum liegen (das bedeutet im ersten Knoten liegen nur Ziele ohne Vorgänger). Somit wird die Ornung der Ziele durch die Vater-Sohn-Beziehung repräsentiert. Dann wird versucht, Mz mit einem der schon vorhanden Söhne zu unifizieren. Gelingt dies, so wird in diesem Ast fortgefahren. Gelingt es nicht, wird ein neuer Knoten angelegt und dort fortgefahren.

5. Jetzt werden für jeden Knoten im Zielteilbaum die jeweils relevanten Prädikate des Startzustandes in die jeweiligen Bäume zur Unterscheidung des Startzustandes einsortiert. Dies geschieht, indem in jeder Stufe des Baumes der Nachfolgeknoten ausgewählt wird, bei dem die größte Überlappung vorliegt. Das ist der Knoten, bei dem die größte Anzahl von Prädikaten unifiziert werden konnte. Nun werden drei Fälle unterschieden (siehe Abb. 4.1). Sei X die Menge der Prädikate des Knotens mit der maximalen Überlappung und Y die Menge der noch einzufügenden Prädikate des neuen Falls.

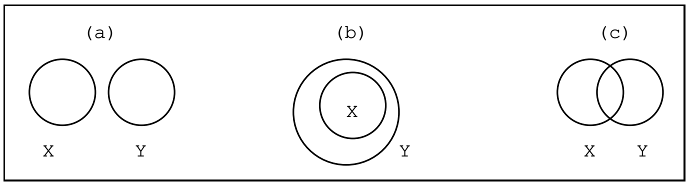

Abbildung 4.1: Situationen beim Einfügen

(a) Es wurde kein Knoten gefunden, mit dem eine echte Überlappung vorliegt. Dann wird ein neuer Knoten erzeugt, der dann alle restlichen, noch einzufügenden Prädikate enthält und der mit dem einzufügenden Fall markiert wird. Das Einfügen ist damit beendet.

(b) Es konnten alle Prädikate in dem Knoten mit der maximalen Überlappung unifiziert werden. Dann wird dieser mit dem einzufügenden Fall markiert und mit den restlichen Prädikaten bei den Nachfolgern fortgefahren.

(c) Es konnten nicht alle Prädikate in dem Knoten A mit der maximalen Überlappung unifiziert werden. Dann wird ein neuer Knoten B erzeugt, der Nachfolger von A wird. Die gemeinsamen Prädikate bleiben in A, die übrigen werden aus A entfernt und in B eingefügt. A wird mit dem einzufügenden Fall markiert und mit seinen Nachfolgern fortgefahren.

Bei dieser Vorgehensweise wird also immer nur bezüglich der nächsten Stufe optimal einsortiert. Nach Einfügen mehrerer Fälle könnte durch eine komplette Umstrukturierung der einzelnen Bäume zur Unterscheidung des Startzustandes evtl. die Gesamtknotenanzahl oder die Tiefe reduziert werden, was eine leichte Effizienzsteigerung bedeuten würde. Die Anzahl der Fälle in einem solchen Baum ist aber durch die vorhergehenden Stufen in der Regel nicht sehr groß und die Anzahl der Prädikate des Startzustandes wird durch die Aufteilung, in nur für einen Zielknoten relevante, auch stark reduziert. In Domänen, in denen diese Teilbäume trotzdem groß werden, kann sich eine solche Umstrukturierung lohnen.

4.2 Beispiele

Das Einfügen in die ersten drei Stufen ist relativ einfach und kann anhand der Beispiele in Kapitel 3 nachvollzogen werden. Zum besseren Verständnis für das Einfügen der Stufen vier und fünf folgen zwei Beispiele :

Zur Verdeutlichung des Einfügens in die 4. Stufe wird das Einfügen von Fall *2_Tuerme_C* in die oben aufgebaute Fallbasis betrachtet, siehe Abb. 4.2 (a). Vorher existierten nur die Äste *2_Tuerme_A* und *2_Tuerme_B* in dem Teilbaum. Beim Einfügen von Fall *2_Tuerme_C* in Stufe vier wird bei Knoten *K1* mit den Söhnen *K2* und *K3* begonnen. *Mz* ist im ersten Schritt $\{On(d,t)\}$. Diese kann nicht mit *K3* aber mit *K2* unifiziert werden, also wird bei *K2* fortgefahren. *Mz* ist jetzt $\{On(c,d)$, $On(a,b)\}$, welche mit *K4* nicht unifizierbar ist. Also wird ein neuer Knoten *K5* mit den Prädikaten von *Mz* erzeugt, und der Einfügevorgang der vierten Sufe ist beendet, da alle Ziele eingefügt wurden.

Zur Verdeutlichung des Einfügens in die 5. Stufe wird das Einfügen von Fall *Turm_3_B* in die oben aufgebaute Fallbasis betrachtet, siehe Abb. 4.2 (b). Vorher existierten nur die Knoten *K1*, *K2* und *K4* in den Teilbäumen zur Unterscheidung des Startzustandes, wobei in *K4* zu dem Zeitpunkt auch noch $Clear(a)$ steht. Beim Einfügen der relevanten Prädikate in den Teilbaum des Zielknotens $On(c,t)$ tritt Fall (b) von Stufe 5 ein, also muß nur *K1* mit 2. markiert werden. Beim Zielknoten $On(b,c)$ tritt Fall (a) ein, also muß *K3* erzeugt und markiert werden. Beim Zielknoten $On(a,b)$ tritt Fall (c) ein, also wird *K5* erzeugt. In *K4* bleibt das gemeinsame Prädikat und *K4* wird mit 2. markiert, während $Clear(a)$ nach *K5* geschoben wird.

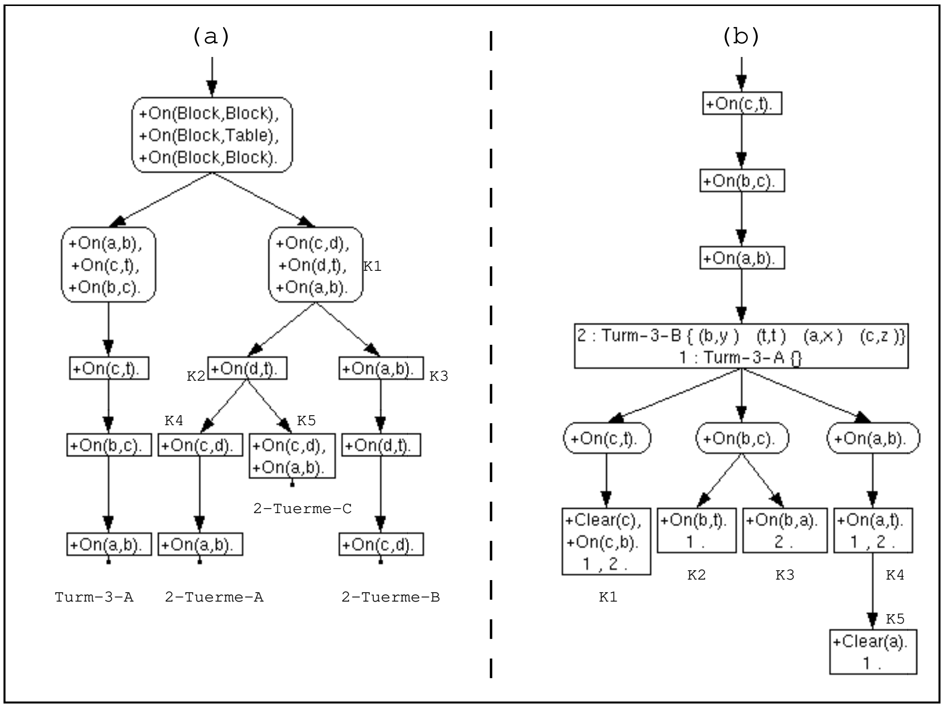

Abbildung 4.2: (a) Einfügen in die 4. Stufe (b) Einfügen in die 5. Stufe

4.3 Bemerkungen

Das Einfügen in den Zielteilbaum ist in 4.1 Punkt 4. für ein besseres Gesamtverständnis etwas vereinfacht dargestellt. Hier die genaue Beschreibung :
In der 4. Stufe gibt es oft mehere Möglichkeiten, die einzelnen Ziele zu unifizieren. Daher wird zuerst ein optimaler Ast zum Einfügen gesucht (der, bei dem die meisten Ziele unifiziert werden können), und dann wird der Fall dort eingefügt. Dabei muß aber die Zielreihenfolge zwischen vorhandenem Ast und einzufügenden Fall völlig identisch sein (im Gegensatz zur Fallauswahl, siehe 5.2.1).
Die Suche nach dem optimalen Ast erfolgt, indem bei jedem Knoten alle möglichen Unifikationen getestet werden (siehe folgendes Beispiel). Dabei wird frühzeitig abgebrochen, falls schon alle Ziele unifiziert wurden. Ist ein optimaler Ast gefunden, so wird, wie oben beschrieben, dort der Fall eingefügt.

Beispiel für das optimale Einfügen in den Zielteilbaum :
Es sei die in Abb. 4.3 dargestellte Situation beim Einfügen im Zielteilbaum vorhanden. Werden die Bindungen $\{(a,x),(b,y)\}$ in der 1. Ebene gewählt, so kann in der zweiten Ebene *K6* weder mit *K2* noch mit *K3* unifiziert werden. Werden aber $\{(a,y),(b,x)\}$ gewählt, so kann in der 2. Ebene *K6* mit *K3* unifiziert werden mit zusätzlicher Bindung (c,z). Also wird diese Möglichkeit gewählt, da insgesamt mehr Ziele unifiziert werden können. In der 3. Ebene muß dann ein neuer Knoten eingefügt werden, da *K7* nicht mit *K4* unifiziert werden kann.

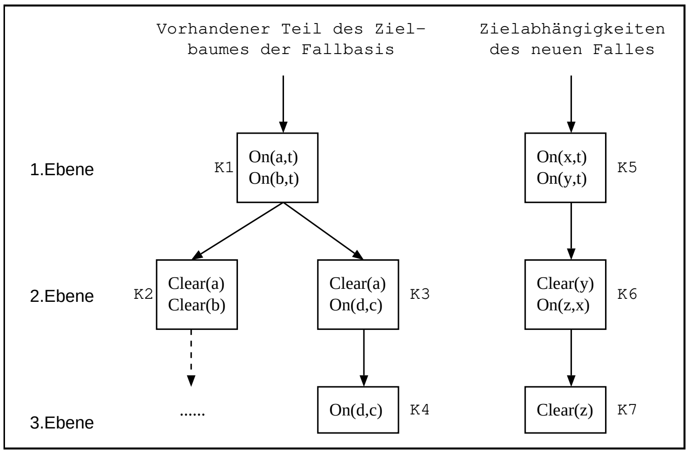

Abbildung 4.3: Komplexes Einfügen in die 4. Stufe

Kapitel 5

Fallauswahl in CAPlan/CbC

5.1 Allgemeines zur Fallauswahl

Ein wichtiger Unterschied zwischen der Architektur der Fallbasis von Prodigy/Analogy und der von CAPlan/CbC ist die Berücksichtigung der Zielabhängigkeiten durch Einfügen des Zielteilbaumes, der die Zielordnung wiederspiegelt. Daher wird das Laufzeitverhalten der Fallauswahl und die Güte der ausgewählten Fälle verbessert, wenn auch bei einem neuen Problem die Reihenfolge der Ziele (teilweise) bekannt ist. In der Tat gibt es domänen-abhängige Methoden, mit denen man zumindest einen Teil der vorhandenen Zielordnung feststellen kann, ohne daß man die Lösung des Problems kennt [5]. Also wird ein neues Problem zusammen mit einer partiellen Ordnung der Ziele, die auch leer seinen darf, als Eingabe der fallbasierten Komponente gegeben. Als Ausgabe wird dann eine Menge von Fällen, die auch leer sein kann, zurückgegeben. Diese Fälle decken zusammen das neue Problem möglichst weit ab. Zu jedem ausgewählten Fall werden die Ziele angegeben, die einen Teil des neuen Problems abdecken und die Ziele des neuen Problems, die von ihm abgedeckt werden und die daraus resultierenden Bindungen der Variablen.

5.2 Beschreibung des Hauptalgorithmus

Die Unterteilung des neuen Problems in seine zusammenhängende Komponenten ist vor der Lösung nicht bekannt, sie ist allerdings durch die evtl. vorhandene Zielordnung eingeschränkt (siehe Ende dieses Absatzes). Deckt man das neue Problem mit vielen Fällen, die jeweils wenig Ziele abdecken, ab, wobei aber die Ziele des neuen Problems interagieren, so ist die gewonnene Information für das Planungssystem gering, da es z.B. keine Information über die Reihenfolge der Erfüllung der Ziele erhält. Der umgekehrte Fall ist weniger kritisch, denn interagieren die Ziele nicht, und es wird ein Fall ausgesucht, der alle Ziele abdeckt, so resultiert daraus höchstens eine zu starke Ornung. Eine sinnvolle Strategie ist es also, das neue Problem mit möglichst wenig Fällen abzudecken.

Der nun beschriebene Algorithmus verfolgt dieses Ziel mit einer Greedy-Methode.
Sei FINDE-ÄHNLICHEN-FALL (PROBLEM, ANZAHL) ein Algorithmus, der in der Fallbasis nach einem zu PROBLEM ähnlichen Fall sucht, der ANZAHL viele Ziele von PROBLEM abdeckt und der, wenn ein solcher gefunden wurde, diesen zurückgibt (wird später beschrieben). Sei P das neue Problem, sei Nz die Anzahl der Ziele von P und Na die Anzahl der Ziele, die von einem Fall abgedeckt werden sollen. Der Hauptalgorithmus zur Fallauswahl durchläuft folgende Schritte:
Zuerst wird versucht, das neue Problem mit einem Fall abzudecken, also ist Na gleich Nz (mit Aufruf von FINDE-ÄHNLICHEN-FALL (P, Na)). Gelingt dies nicht, so wird Na solange dekrementiert und erneut versucht einen ähnlichen Fall zu finden, bis ein solcher gefunden wurde oder Na gleich Null ist. Wenn ein Fall gefunden wurde, wird mit den restlichen Zielen, also denen, die noch nicht abgedeckt wurden, fortgefahren (P stellt dann nur noch ein Teilproblem des ursprünglichen dar). Ist Na gleich Null oder wurden alle Ziele abgedeckt, so werden die gefundenen Fälle mit den oben genannten Zusatzangaben zurückgegeben.
Bei dieser Vorgehensweise muß allerdings eine Bedingung eingehalten werden. Wird ein Ziel von einem Fall abgedeckt, so müssen auch alle Vorgänger dieses Zieles bezüglich der angegebenen Zielordnung vom gleichen Fall abgedeckt werden, da sie auf jeden Fall zu einer zusammenhängenden Komponente gehören. Also schränken solche Ordnungen die Aufteilungsmöglichkeiten der Ziele stark ein.

5.2.1 Beschreibung von FINDE-ÄHNLICHEN-FALL

Wie beim Einfügen eines Falles erfolgt die Fallauswahl gemäß des Aufbaus des Index in fünf Schritten. Hier sind allerdings im Gegensatz zum Einfügen nicht nur in der 4. Stufe mehrere Äste vorhanden (vgl 4.3), die untersucht werden müssen (in deren Teilbäumen ähnliche Fälle liegen können). Dies geschieht durch Tiefensuche mit Backtracking.

1. Es wird der Knoten mit ANZAHL vielen interagierenden Zielen ausgewählt.

2. Da Nz größer als ANZAHL seien kann, muß hier ein Teilmengentest erfolgen. Bei allen Knoten, in denen die Menge der klassenparametrisierten Ziele eine Teilmenge aller klassenparametrisierten Ziele von PROBLEM ist, muß weitergesucht werden.

3. In dieser Ebene wird bei allen Nachfolgern weitergesucht, da die Unifizierbarkeit der einzelnen Ziele zusammen mit der Zielordnung getestet wird. Diese Stufe dient also nur der Übersichtlichkeit für den Benutzer.

4. Bei den Zielabhängigkeiten muß berücksichtigt werden, daß die schon vorgegebene Ordnung der Ziele des neuen Problems eine Teilordnung eines potentiell ähnlichen Falles sein muß. Es dürfen also Fälle mit strikterer Ordnung (da die

partielle Ordnung des neuen Problems ja nicht vollständig sein muß) aber nicht mit schwächerer Ordnung herangezogen werden. Dieser Test geschieht schrittweise in jeder Stufe des Zielteilbaumes. In jedem Schritt wird dazu wie beim Einfügen die Menge Mz der Ziele von PROBLEM bestimmt, deren Vorgänger bereits in den Vaterknoten des Zielteilbaumes liegen. Nun muß bei allen Söhnen, bei denen nach Unifikation die Ziele eine Teilmenge von Mz sind, weitergesucht werden. In jedem Schritt werden die bereits abgedeckten Ziele markiert und die bei den Unifikationen entstehenden Variablenbindungen aufgesammelt. Diese werden dann in den nächsten Stufen berücksichtigt. Konnten insgesamt ANZAHL viele Ziele unifiziert werden, so wurde durch die Wahl in der 1. Stufe automatisch ein Blatt des Zielteilbaumes erreicht.

5. Wenn man bei den Prädikaten des Startzustandes ebenfalls eine völlige Übereinstimmung fordern würde, könnten (bis auf die angegebene Zielreihenfolge) nur wirklich gleiche Fälle ausgewählt werden. Dies ist in der Regel zu strikt. Daher verlangt man meistens nur eine teilweise Übereinstimmung. Diese wird hier durch eine Rate R festgelegt, welche in Prozent den Anteil von allen Prädikaten des Startzustandes des Falles angibt, der mindestens übereinstimmen muß.
   Bei der Auswahl in dieser Stufe wird zunächst jeder Fall, der in diesem Teilbaum liegt, markiert, und für jeden Fall wird die Anzahl der unifizierbaren Prädikate Pos und der nicht unifizierbaren Prädikate Neg mitgefürt. Es werden die einzelnen Teilbäume zur Unterscheidung des Startzustandes bezüglich der jeweiligen Ziele solange durchsucht, bis entweder kein Fall mehr markiert ist (es konnte kein ähnlicher Fall gefunden werden), oder bis ein Fall der Rate R genügt. Dazu wird zu jedem Knoten die Menge Uni der Prädikate von diesem Knoten bestimmt, die unter der Berücksichtigung der bisher entstandenen Variablenbindungen mit den Prädikaten des Startzustandes von PROBLEM unifiziert werden können. Für jeden Fall, der in diesem Knoten aufgeführt ist werden Pos und Neg bezüglich Uni aktualisiert. Da zusätzlich die Gesamtzahl der Prädikate Ges der Startzustände der Fälle bekannt ist, kann nach jedem Schritt durch Vergleich von Pos, Neg und Ges getestet werden, ob der Fall R bereits genügt (dann wird abgebrochen), und es kann festgestellt werden, ob der Fall überhaupt noch R genügen kann (wenn nicht, wird er demarkiert). In Ästen, in denen keine markierten Fälle mehr vorhanden sind, wird nicht mehr weitergesucht.

5.2.2 Bemerkungen

1. Beim Hauptalgorithmus wird standartmäßig kein Backtracking durchgeführt. Dazu ein Beispiel: Nz sei 10. Mit Na gleich 10, .., 7 war die Suche erfolglos, aber mit Na gleich 6 wurde ein Fall gefunden, die restlichen 4 Ziele konnten jedoch nicht mehr abgedeckt werden. Damit ist die Suche beendet. Nun können in der Fallbasis jedoch zwei Fälle mit jeweils 5 Zielen vorhanden sein, die zusammen

alle 10 Ziele abdecken. Diese könnten durch Backtracking, da ja die letzten 4 Ziele nicht abgedeckt wurden, gefunden werden. Dies ist noch nicht implementiert. Außerdem bräuchte man dazu eine Entscheidungsprozedur für die beste Aufteilung im allgemeinen Fall. Ist für 10 Ziele z.B. eine 6-2-2 Aufteilung oder eine 5-5 Aufteilung besser ?

2. Wird ein Fall mit Na vielen Zielen gefunden und ist Np die Anzahl der Ziele von P, die noch nicht abgedeckt ist und gilt $Np < Na$, dann wird Na direkt auf Np gesetzt, da größere Na keinen Sinn machen. Ist aber $Np \geq Na$, dann wird FINDE-ÄHNLICHEN-FALL nochmal mit Na aufgerufen, um evtl. einen weiteren Fall mit der gleichen Zielanzahl zu finden (möglicherweise sogar nochmal den gleichen). Allerdings wird direkt an der Stelle, wo der letzte Fall gefunden wurde, mit der Suche fortgefahren, da vorher kein weiterer mehr gefunden werden kann. Dazu werden der letzte durchsuchte Pfad im Baum und die letzten Bindungen immer mitgeführt.

3. Es gibt auch Domänen, in denen eine inverse Strategie sinnvoller sein kann. Dabei versucht man ein neues Problem mit möglichst vielen kleinen zusammenhängenden Komponenten abzudecken. In einer nächsten Version wird eine solche Strategie angeboten, die über die Konfiguration einstellbar sein wird.

4. Beim Einfügen in die 4. Stufe muß in jedem Schritt die Menge der Ziele des zu untersuchenden Knotens nur eine Teilmenge von Mz sein (das entspricht der Tatsache, daß die Ordnung des alten Falles strikter sein darf). Daher ist es möglich, daß es mehrere Kombinationen von Zielen gibt, mit denen man bei einem Knoten fortfahren muß. Dabei können unterschiedliche Variablenbindungen entstehen, dazu ein Beispiel:
Sei $Mz = \{On(a,b), On(b,c), On(d,e)\}$ und enthalte der zu untersuchende Knoten die Ziele $\{On(w,x), On(y,z)\}$, dann sind folgende Kombinationen möglich: ($A - B$ bedeutet hier, daß A mit B unifiziert wird)

   1. $On(a,b) - On(w,x)$, $On(d,e) - On(y,z)$

   2. $On(a,b) - On(y,z)$, $On(d,e) - On(w,x)$

   3. $On(b,c) - On(w,x)$, $On(d,e) - On(y,z)$

   4. $On(b,c) - On(y,z)$, $On(d,e) - On(w,x)$

   Mit all diesen Kombinationen muß mit den entsprechenden Bindungen in der nächsten Ebene fortgefahren werden. Hier wird ein großer Vorteil der Berücksichtigung der Zielabhängigkeiten deutlich: Ohne diese Berücksichtigung müssen alle Ziele des neuen Problems mit allen Zielen des zu untersuchenden Falles verglichen werden. Dagegen müssen mit Zielabhängigkeiten immer nur kleine Teilmengen unifiziert werden. Dies ist ein großer Vorteil, da die Anzahl der möglichen Unifikationen exponentiell in der Anzahl der Ziele oder in der Anzahl der Variablen steigen kann.

5. Es gibt Domänen, in denen die Zielabhängigkeiten nicht transitiv sind (z.B. die Domäne für mechanische Werkstücke). Dort ist es wichtig, daß ein Ziel direkt vor einem anderen erfüllt wird. Ein Fall mit der Abhängigkeit A->B->C dürfte nicht für ein neues Problem mit der Abhängikeit A->B, A->C verwendet werden. Diese Art der Abhängigkeiten kann in der Konfiguration der fallbasierten Komponente eingestellt werden und wird in der 4. Stufe entsprechend berücksichtigt.

5.3 Ein Beispiel

Sei *2_Tuerme_D* ein neues Problem P mit dem Startzustand {*On(a,e), Clear(a), On(e,t), On(c,t), Clear(c), On(d,b), Clear(b), On(b,t)*} und folgendem Zielzustand {*On(a,b), On(c,d), On(d,t), On(b,t)*}. Mit der Zielabhängigkeit *On(d,t) –> On(a,b)* wird dieses an die oben aufgebaute Fallbasis gegeben. Bei einer Rate $R = 80\%$ entsteht folgender Ablauf:
Zuerst wird versucht, einen Fall mit vier Zielen zu finden, da $Nz = 4$. Dieser Test scheitert bereits in der 1. Stufe, da überhaupt kein Fall mit vier Zielen in der Fallbasis enthalten ist. Also wird *Na* auf drei gesetzt. In der 1. Stufe wird der Ast mit 3 Zielen ausgewählt (siehe Abb. 3.3). In der 2. Stufe ist die Menge mit den drei Zielen des einzigen Knotens eine Teilmenge der klassenparametrisierten Form der Ziele von P, also wird damit fortgefahren. In der 3. Stufe werden alle Nachfolger untersucht. In der 4. Stufe (siehe Abb. 3.4 (a)) können im Ast *Turm_3_A* die Ziele der ersten beiden Knoten mit den Zielen von P unifiziert werden, aber nicht mehr *On(a,b)*, daher wird dort abgebrochen. Im Ast *2_Tuerme_B* widerspricht die Zielordnung der Zielabhängigkeit von P, daher wird beim zweiten Knoten abgebrochen. Dagegen können alle Ziele der beiden Äste *2_Tuerme_A* und *2_Tuerme_C* unifiziert werden, wobei das Ziel *On(b,t)* von P übrigbleibt. Außerdem ist die Zielabhängigkeit von P eine Teilmenge der Ordnung, die durch die beiden Äste gegeben ist. Also wird jeweils mit den Bindungen $\{(a,a),(b,b),(c,c),(d,d)\}$ in der 5. Stufe fortgefahren.
Test des Startzustandes in der 5. Stufe von *2_Tuerme_A* (siehe Abb. 5.1): In *K1* kann nur Clear(d) unifiziert werden. In *K2* kann *On(c,b)* mit keinem Prädikat von P unifiziert werden. Da *2_Tuerme_A* insgesamt fünf relevante Prädikate besitzt und schon zwei nicht abgedeckt werden können, wird hier bereits abgebrochen, da R nicht mehr erreicht werden kann.
Test des Startzustandes in der 5. Stufe von *2_Tuerme_C* (siehe Abb. 5.1): In *K4* können beide Prädikate unifiziert werden, also muß *K5* noch untersucht werden. Dort können alle bis auf On(a,t) mit Prädikaten von P unifiziert werden, so daß insgesamt 83% abgedeckt werden, was R genügt.

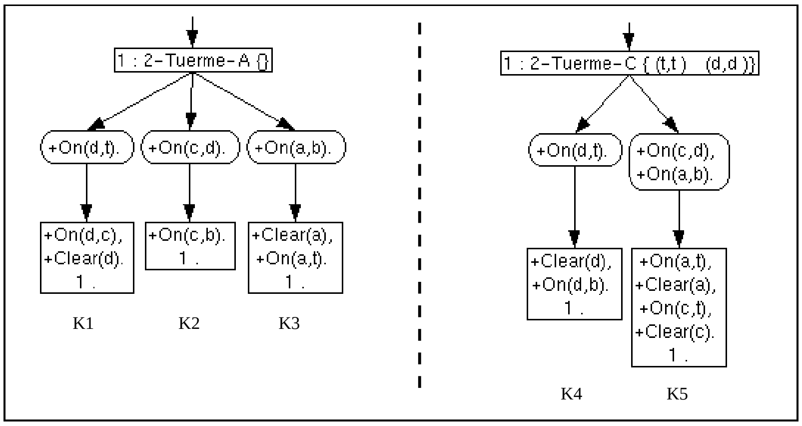

Abbildung 5.1: Auswahl in der 5. Stufe

Demnach wurde der Fall *2_Tuerme_C* gefunden, der drei der vier Ziele von P abdeckt. Nun wird *Na* auf eins gesetzt und das letzte Ziel *On(b,t)* noch untersucht. Es wird der Ast mit einem Ziel ausgewählt. Allerdings ist die klassenparametrisierte Form von *On(b,t)*, nämlich *On(Block,Table)*, zu keiner der in der 2. Stufe stehenden gleich, und somit wird schon hier abgebrochen.
Insgesamt wird dann der Fall *2_Tuerme_C* mit den drei Zielen, die er abdeckt, und den Variablenbindungen zurückgegeben.

5.4 Bemerkungen

Es gibt mehrere Möglichkeiten, den Test, ob ein Fall F für ein neues Problem P einer Rate R genügt, durchzuführen:

(a) Es müssen mindestens R % viele Prädikate von F mit Prädikaten von P unifizierbar sein. Dieser Test ist implementiert und ist für die Domäne Klötzchen-Welt sinnvoll.

(b) Es müssen mindestens R % viele Prädikate von P mit Prädikaten von F unifizierbar sein. Dieser Test ist für die Domäne der mechanischen Werkstücke sinnvoller.

(c) Es müssen mindestens R % viele Prädikate von P und F unifizierbar sein, bezogen auf die Vereinigung der Prädikate von P und F. Dieser Test stellt einen Kompromiß dar.

Die Möglichkeiten (b) und (c) sollen in einer nächsten Version angeboten werden.

Kapitel 6

Kurzbeschreibung der Benutzeroberfläche in CAPlan/CbC

Fall Einfügen: Nachdem ein Problem mit dem *CAPlan System Browser* gelöst wurde, kann dieses zusammen mit dem erstellten Plan als neuer Fall eingefügt werden, indem man im Browser *CAPlan/CbC* im Menü *Browsers* den Eintrag *Create Case* anwählt. In einem Dialogfenster kann man dem Fall dann noch einen Namen geben.

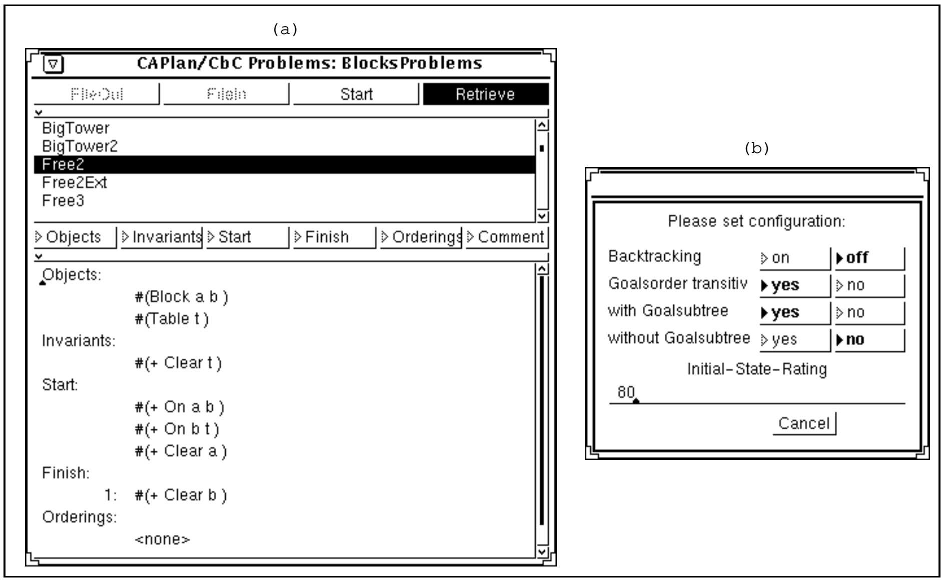

Abbildung 6.1: (a) Fallauswahl (b) Konfigurationsmenü

Fallauswahl: Nach Auswahl eines Problems im Browser *CAPlan/CbC Problems*, wird die Suche nach ähnlichen Fällen durch Anwählen von *Retrieve* im Browser *CAPlan/CbC Problems* gestartet.

Konfiguration: In das Konfigurationsmenü gelangt man durch anwählen von *Configuration* im Browser *CAPlan/CbC*. Dort sind einstellbar:

1. Das Backtracking für den Hauptalgorithmus der Fallauswahl (ist noch nicht implementiert).
2. Transitivität der Zielabhängigkeiten in der 4. Stufe bei der Fallauswahl.
3. Das Anlegen einer zusätzlichen Fallbasis ohne Zielabhängigkeiten (die 4. Stufe entfällt) zum Vergleich.
4. Die Rate R für den Vergleich der Startzustände bei der Fallauswahl.

Index der Fallbasis: Der Baum der Fallbasis kann durch Anwählen von *Index* im Browser *CAPlan/CbC CaseBase* angeschaut werden. Im *NodeMenu* können Methoden angewählt werden, die u.a. Teilbäume ein- oder ausblenden.

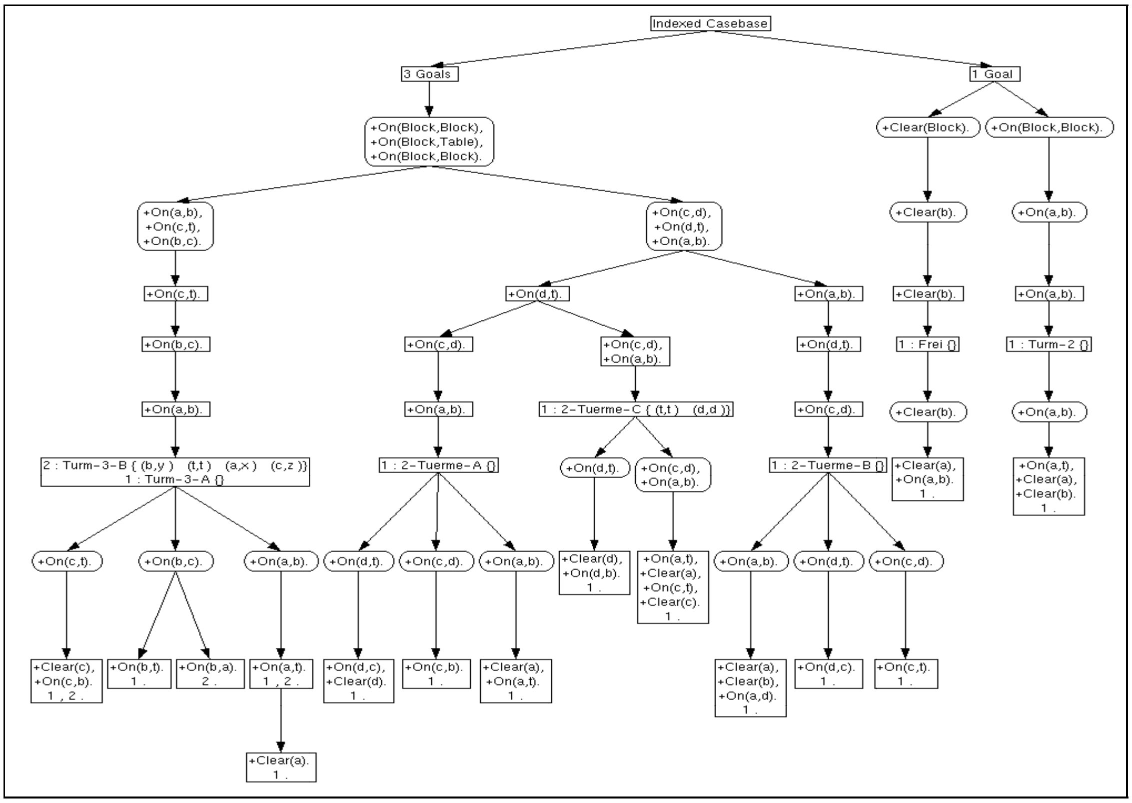

Abbildung 6.2: Vollständige Struktur der aufgebauten Fallbasis

Kapitel 7

Implementierung

Das System wurde in Smalltalk-80 [1] implementiert.

Im folgenden Überblick sind die wichtigsten implementierten Smalltalkklassen, aufgeführt, sie befinden sich in der Klassenkategorie *CAPIndexedCasebase*. Der Index selber besteht aus Instanzen dieser Klassen, die den unterschiedlichen Stufen und damit unterschiedlichen Knotenformaten entsprechen. Sie enthalten jeweils die nötigen Instanzvariablen und u.a. Methoden für das Einfügen und Auswählen von Fällen.

Stufe	Name	Beschreibung
0	CaseBaseIndexed	stellt Wurzel des Index dar
1	InteractionList	Knoten der ersten Stufe
2,3	GoalLink	Knoten der zweiten und dritten Stufe
4	GoalNode	Knoten der vierten Stufe
5a	InitialStartNode	Wurzel der Bäume in der fünften Stufe mit allen Informationen über die Fälle dieses Teilbaumes
5b	InitialNode	übrige Knoten der fünften Stufe

Überblick der Methoden für das Einfügen von Fällen in den einzelnen Stufen:

Stufe	Name	Bemerkungen
0	*addCase:*	-
1	*addComponent:number:case:caseBase:*	-

2	*addLink:case:caseBase*	-
3	*updateStateNet:case:*	-
4	*testComponent:allAdded:newAdded:nodeAdded: nodeBindings:order:modulo:optimum:current:*	Suchen der optimalen Bindungen
	addComponent:case:modulo:optimum: depth:allAdded:	Einfügen in vorhandenen Ast
	addComponentNew:case:modulo:optimum: depth:allAdded:	Aufbau neuer Knoten
5a	*addStateNet:case:modulo:allBindings:optimum:*	-
5b	*addStateNet:case:modulo:caseBindings:number:*	-

Überblick der Methoden für die Fallauswahl in den einzelnen Stufen:

Stufe	Name
0	*retrieve:rating:*
1	*findSimilarCase:order:rating:goalsNumber:path:connections:predecessors:*
2	*findSimilarCase:order:rating:goalsNumber:path:connections:predecessors:*
3	*findSimilarCase:order:rating:goalsNumber:path:connections:predecessors:*
4	*testRetrieve:allAdded:newAdded:nodeAdded:nodeBindings:modulo: goalsNumber:rating:path:connections:predecessors:lastGoals:neededGoals:*
5a	*testInitialState:rating:modulo:*
5b	*testInitialState:rating:modulo:cases:*

Bemerkung: Zusätzlich sind für einige dieser Methoden Varianten für den Vergleich mit einem Index ohne vierte Stufe vorhanden. Deren Namen haben den Zusatz *Vergleich*.

Kapitel 8

Schlußbemerkungen

Es wurde mit der gleichen Menge von Fällen der Domäne der Arbeitsplanung jeweils eine Fallbasis mit der Struktur von Prodigy/Analogy und von CAPlan/CbC aufgebaut. Beim Vergleich der Retrieval-Zeiten stellte sich heraus, daß bei großen Zielanzahlen die Berücksichtigung der Zielabhängigkeiten wesentlich geringere Retrieval-Zeiten zur Folge hatte [5]. Die Verbesserung des Laufzeitverhaltens hat zwei Gründe:

1. Das kombinatorisch bedingte exponentielle Wachstum des Aufwandes mit der Anzahl der Ziele wird durch die Berücksichtigung der Zielordnung abgeschwächt. Je stärker die Ordnung ist, desto größer ist die Reduzierung der Unifikationsmöglichkeiten.

2. Der Vergleich der Startzustände ist effizienter, da zusätzlich eine Abhängigkeit zwischen den Zielen und den jeweils zugehörigen Prädikaten des Startzustandes betrachtet wird und somit weniger Unifikationsmöglichkeiten entstehen.

Weiterhin werden durch die Berücksichtigung der Abhängigkeiten zwischen den Zielen bessere (ähnlichere) Fälle ausgewählt.
Die in der vorgestellten Architektur enthaltene Information der Abhängigkeiten zwischen den Zielen und den jeweils relevanten Prädikaten des Startzustandes wird noch nicht optimal ausgenutzt. Durch domänenabhängige Methoden, die für ein ungelöstes Problem solche Abhängigkeiten bestimmen, könnte eine weitere Verbesserung erzielt werden.

Literaturverzeichnis

[1] Shafer, D., Ritz, D.A. (1991). *Practical Smalltalk.* Springer-Verlag.

[2] Muñoz, H., Paulokat, J., Wess, S. (1995). *Controlling Nonlinear Hierarchical Planning by Case Replay.* In Mark Keane, Jean Paul Haton, Michel Manago (Hrsg.), Topics in Case-Based Reasoning, LNAI series.

[3] Veloso, M. M. (1992). *Learning by Analogical Reasoning in General Problem Solving.* CMU-CS-92-174, School of Computer Science, Carnegie Melon University, Pittsburg, PA 15213.

[4] Weberskirch, F. (1994). *Realisierung eines nichtlinearen Planungssystems zur Unterstützung der Arbeitsplanerstellung bei der computerintegrierten Fertigung (CIM).* Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern.

[5] Muñoz, H., Hüllen, J. (1995) *Retrieving Cases in Structured Domains by Using Goal Dependencies.* In Manuala Veloso, Agnar Aanodt (Hrsg.), International Conference on Case-Based Reasoning, Topics in Cases-Base Reasoning, LNAI series.

[6] Kettnaker, V. (1995) *Konzeption und Realisierung einer Toolbox statischer Kontrollmethoden zur Steuerung eines Causal Link Planers.* Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern.