Strukturorientiertes fallbasiertes Schließen

Dissertation

zur Erlangung des akademischen Grades

Doktor-Ingenieur

an der Fakultät Bauingenieurwesen

der

Bauhaus Universität Weimar

vorgelegt von

Carl-Helmut Coulon

aus Bonn

Weimar

Gutachter: 1. Prof. R. Hübler

2. Prof. H. D. Burkhard

3. Dr. A. Voß

Tag der Disputation: 21. November 1997

Seitenanzahl: 160

Thesen dieser Arbeit:

Problemstellung

1. Das zur **Erstellung eines Bauplans** benötigte Wissen ist extrem komplex, auf Experten mehrerer, sich gegenseitig beeinflussender Teilbereiche verteilt und teilweise unvollständig und widersprüchlich. Deshalb ist die **Akquisition** und Modellierung von **Regelwissen** zum Zweck der Computerunterstützung von Entwurfsprozessen in diesem Arbeitsbereich **sehr arbeitsintensiv, wenig erfolgversprechend und bleibt auf kleine Teilbereiche beschränkt**.

2. Bei der computergestützten Erstellung von Bauplänen entsteht **Erfahrungswissen** in Form von digitalen Bauplänen. Dazu wird **kein zusätzlicher Aufwand** benötigt. Dieses Erfahrungswissen wird mit der Zeit alle Teilbereiche der Bauplanererstellung überdecken und bildet damit **potentielles Wissen zur Unterstützung** der Bearbeitung von ähnlichen Problemen.

3. Bei der **Erstellung von Bauplänen** sind **strukturelle Abhängigkeiten** zwischen Datenobjekten, zum Beispiel räumliche Beziehungen, **wichtig**. Diese Abhängigkeiten können in **Strukturgraphen** abgebildet werden.

4. Der **Vergleich zweier Strukturgraphen** zur Feststellung von Ähnlichkeit besitzt im allgemeinen **exponentielle Komplexität** in Abhängigkeit von der Größe der Graphen.

5. Der Mangel an Regelwissen in Kontrast zu vorhandenem Erfahrungswissen und die Bedeutung struktureller Abhängigkeiten **findet sich in vielen ähnlich komplexen Anwendungsbereichen wieder**, wie

zum Beispiel dem Entwurf chemischer Moleküle zur pharmazeutischen Verwendung.

Bisherige Lösungsansätze

1. Es gibt **keinen regelbasierten Ansatz** zur **flächendeckenden Unterstützung** der **Erstellung von Bauplänen**. Da das Regelwissen auf Teilbereiche beschränkt ist, wird die Anwendbarkeit jedes regelbasierten Ansatzes auf die modellierten Teilbereiche eingeschränkt.

2. Das **fallbasierte Schließen nutzt Erfahrung** in Form von Fällen zur Lösung von in Anfragen enthaltenen Problemen. Im fallbasierten Schließen wird ein zu einer Anfrage ähnlicher Fall gesucht und Information aus diesem auf die Anfrage übertragen.

3. Zur Übertragung und **Anpassung** von Information aus einem Fall auf eine Anfrage gibt es bisher **kein generisches Konzept**, sondern einige auf eingeschränkte Bereiche anwendbare Lösungen. Bisherige Konzepte des fallbasierten Schließens spezifizieren nicht, oder nur teilweise, das von ihnen **benötigte anwendungsspezifische Regelwissen**. Auf einzelne Anwendungsbereiche spezialisierte Ansätze benötigen insbesondere zur Übertragung und Anpassung anwendungsspezifisches Regelwissen, welches im allgemeinen auf Teilbereiche beschränkt ist. Dadurch wird auch die Anwendbarkeit dieser fallbasierten Ansätze auf Teilbereiche eingeschränkt, obwohl das Erfahrungswissen weit mehr abdeckt. Insbesondere sind bisherige **Ansätze zur Anpassung von Bauplänen auf extrem kleine Teilbereiche beschränkt**.

4. Die meisten fallbasierten Verfahren arbeiten auf **Attributwerten** und **nicht auf Strukturen** und nur wenige Verfahren formulieren den **Zusammenhang zwischen Ähnlichkeit und Anpaßbarkeit**.

5. Für eingeschränkte Strukturgraphen gibt es Vergleichsverfahren, die polynomielle Komplexität besitzen. Im allgemeinen ist der Vergleich von Strukturgraphen np-vollständig. Es gibt bisher **kein Suchverfahren**, welches **mit nicht exponentiellen approximativen Vergleichsfunktionen** zur Vermeidung expontieller Vergleiche **maximal effizient umgeht**.

6. Es gibt bisher **kein generisches Konzept zur fallbasierten Korrektur** von Anfragen.

Ergebnisse der Arbeit

Annahmen

1. Die **Struktur eines Falles** ist für das fallbasiertes Schließen relevant, hinreichend spezifisch und formulierbar.

2. **Je ähnlicher** sich die Struktur eines Falles und einer Anfrage sind, **desto besser** paßt ein Fall zu einer Anfrage. Fälle, die die Struktur einer Anfrage **komplett enthalten**, sind **problemlos** auf die Anfrage **übertragbar**.

Ansätze und Ergebnisse

3. Das vorgestellte Verfahren kann zum **Finden, Übertragen und Anpassen** von Fällen Regelwissen in Form von **Erkennungs-, Vergleichs-, Übertragungs- und Rekonstruktionsfunktionen** verarbeiten. Es enthält ein **detailliertes generisches Konzept** zum **strukturorientierten fallbasierten Schließen („SFBS")**. Dieses ist exemplarisch am Beispiel der Erstellung **orthogonaler Baupläne** implementiert.

4. Die **Erkennungs- und Rekonstruktionsfunktionen sind anwendungsspezifisch**. Die standardmäßigen **Vergleichs- und Übertragungsfunktionen sind generisch**, können jedoch **anwendungsspezifisch verfeinert** werden. Die Arbeit präsentiert eine Idee zum **Lernen von Verfeinerungen** der Übertragungsfunktionen.

5. Für **einige Anwendungen** werden die **Erkennungs- und Rekonstruktionsfunktionen nicht benötigt**, so daß das Verfahren dort **völlig ohne anwendungsspezifisches Regelwissen anwendbar** ist. Das von meinem Verfahren verarbeitete **anwendungsspezifische Regelwissen** ist **nachweisbar einfacher** zu akquirieren und zu modellieren als das für ein **regelbasiertes System** benötigte Wissen.

6. Die **Erkennungs- und Rekonstruktionsfunktionen erschließen implizit** in den Fällen **enthaltenes Wissen**. Es beschreibt unter an-

derem den Stil dessen, der die Fälle erzeugt hat und ermöglicht auf diese Weise eine stilgerechte Unterstützung.

7. Zur Verarbeitung von **Bauplänen reichen** Erkennungs- und Rekonstruktionsfunktionen für **primitive räumliche Relationen aus**, um Fälle zur Bearbeitung von Anfragen zu benutzen. Diese primitiven räumlichen Relationen repräsentieren relative Entfernungsverhältnisse in Abhängigkeit von der Größe der beteiligten Objekte. **Weitere Erkennungs- und Rekonstruktionsfunktionen** können jedoch Unterschiede zwischen Anfrage und Fall überbrücken und **verbessern** damit die **Qualität des Ergebnisses**.

8. Die **sparsame Verwendung anwendungsspezifischen Regelwissens** macht eine **Übertragung** des vorgestellten Ansatzes **auf andere Anwendungsbereiche attraktiv**.

Eigenschaften des Lösungsansatzes

9. Im vorgestellten Verfahren wird nach dem **Fall mit der größten mit der Anfrage gemeinsamen Teilstruktur gesucht**. Das **Ergebnis** eines Vergleichs ist **keine Zahl**, sondern die **gemeinsame Teilstruktur**.

10. Bei der Suche nach einem geeigneten Fall werden im vorgestellten Verfahren **polynomielle approximative Vergleiche** als Alternative zum exponentiellen Strukturvergleich **maximal ausgenutzt**. Diese approximativen Vergleiche sind auf beliebige Strukturgraphen anwendbar.

11. Der **exponentielle Strukturvergleich** wird zur Übertragung weiterhin benötigt, besitzt jedoch ein **„any-time" Verhalten**. Dies bedeutet, daß während der Laufzeit die Qualität des Ergebnisses monoton ansteigt und das Verfahren jederzeit unterbrochen werden kann. Durch gezielte Vereinfachung der Problemstellung kann der Benutzer die Laufzeit des Vergleichs stark reduzieren.

12. Das in dieser Arbeit vorgestellte Verfahren kann **fallbasiert korrigieren**. Um zu korrigierende Teile einer Anfrage zu bestimmen, wird eine aus der Fallbasis **gelernte Heuristik zur Bewertung** von Anfragen benutzt.

13. Die Anpassung **kombiniert** die Anpassung von **Struktur und Objekteigenschaften**.

Überblick

Die Motivation für diese Arbeit war es, durch Verwendung geeigneter vorhandener Baupläne die Erzeugung neuer Baupläne zu unterstützen. Die Lösung dieser Aufgabe ist Gegenstand der Arbeit. Die Vorgehensweise beruht auf dem Paradigma des fallbasierten Schließens und gehört damit in ein Forschungsgebiet, welches in den letzten Jahren weltweit Beachtung und Förderung fand. Im fallbasierten Schließen werden zu einer Anfrage (wie zum Beispiel für einen neuen Bauplan) Fälle (wie zum Beispiel vorhandene Baupläne) gesucht. Aus den gefundenen Fällen wird Information auf die Anfrage übertragen und gegebenenfalls an die Anfrage angepaßt. In dem Bereich des fallbasierten Schließens gibt es bisher jedoch keinen Ansatz, der den Vergleich, die Übertragung und Anpassung von Bauplänen erlaubt.

Diese Arbeit wird zeigen, daß das zur Verarbeitung von Bauplänen entwickelte Konzept allgemein anwendbar ist und somit erstmalig ein generisches Konzept für das fallbasierte Schließen darstellt, welches auch die Anpassung umschließt. Da in Bauplänen die räumliche Struktur eine wichtige Rolle spielt, ist das Konzept auf strukturorientierte Anwendungen ausgerichtet. Deshalb bezeichne ich es als ein Konzept zum „strukturorientierten fallbasierten Schließen".

Sowohl bei der Erzeugung von Bauplänen als auch bei bisherigen Ansätzen zum fallbasierten Schließen bildet die Bereitstellung des benötigten anwendungsspezifischen Regelwissens das größte Problem. Für die Erstellung von Bauplänen fehlt es größtenteils, und in den bekannten Ansätzen ist nicht genau spezifiziert, wozu welche Art von Wissen benötigt wird. Deshalb werden sich große Teile der Arbeit mit der Spezifikation und Modellierung von Wissen beschäftigen. Es wird dargelegt, welches Wissen zur Suche, Übertragung und Anpassung mindestens benötigt wird, wie das darüber hinausgehende Wissen verarbeitet wird, welche Zusammenhänge es zum Beispiel zwischen Vergleichs- und Anpassungswissen gibt und wie man das Wissen modellieren kann.

Das in der Arbeit vorgestellte Konzept erlaubt die Ergänzung, Detaillierung und Korrektur einer Anfrage. Die beiden entscheidenden Algorithmen dienen dem Vergleich von Anfrage und Fall und der Anpassung der Information des Falles zur Modifikation der Anfrage. Der Schwerpunkt der Algorithmen liegt auf der Effizienz. So wird zum einen ein bekannter Algorithmus zum Vergleich von Strukturen vorgestellt und dessen Effizienz gesteigert. Zum anderen wird ein neuer Constraint-Relaxations-Algorithmus gezeigt,

welcher im Tausch gegen ein erträgliche Einschränkung der Vollständigkeit eine Anpassung mit linearer Komplexität ermöglicht.

Danksagung

Viele Kollegen und Freunde haben mir geholfen, diese Arbeit fertigzustellen. Insbesondere möchte ich meinen Kollegen aus dem Projekt FABEL danken, die immer wieder bereit waren über meine Ideen zu diskutieren.

Zweien von ihnen bin ich besonders zu Dank verpflichtet. Als erstes meinem Zimmerkollegen Jörg W. Schaaf, der zum einen wie kein anderer jederzeit mit mir über beliebig vage Ideen diskutiert hat und zum anderen das Retrieval-Werkzeug ASPECT entwickelt hat, welches mir die Lösung der Retrieval-Aufgabe wesentlich vereinfachte.

Ohne meine Projektleiterin, Frau Dr. Angi Voß, wäre diese Arbeit in dieser Form kaum möglich gewesen. Sie hat mir durch geduldigen Druck geholfen, immer wieder die Arbeit voranzutreiben und gleichzeitig unermüdlich meine Konzepte hinterfragt, bis auch die letzte Unklarheit oder Mehrdeutigkeit beseitigt war.

Besonderer Dank gilt Herrn Prof. Dr. Hübler für die Betreuung der Arbeit aus Sicht der Bauinformatik und seine Bereitschaft sich auf das Thema dieser Arbeit einzulassen. Ebenso möchte ich Herrn Prof. Dr. Burkhard für seine Kommentare zur Arbeit danken, insbesondere für seine Anmerkungen bezüglich der Eigenschaften des Ähnlichkeitsmaßes.

Sankt Augustin im März 1997

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die vorliegende Arbeit beschreibt ein generisches Konzept zur Problemlösung durch Auffinden und Wiederverwenden bekannter Problemlösungen. Diese Vorgehensweise wird in der Informatik als „fallbasiertes Schließen" bezeichnet. Da das Konzept insbesondere auf die Verarbeitung von durch Strukturen repräsentierbaren Problemen und Lösungen ausgerichtet ist, nenne ich es ein Konzept zum „strukturorientierten fallbasierten Schließen" (SFBS[1]). Die Realisierung dieses Konzeptes wird in der Arbeit am Beispiel der Erstellung von Bauplänen und anderen Anwendungen beschrieben.

Bisher gibt es kein generisches Konzept zur Wiederverwendung bekannter Problemlösungen. Ebenso gibt es kein anwendungsspezifisches System zur Erstellung von Bauplänen. In bekannten Ansätzen zum fallbasierten Schließen (vergleiche Kapitel 4 und 8) ist das zur Wiederverwendung benötigte Wissen entweder nicht spezifiziert oder nur für eingeschränkte Anwendungsbereiche geeignet, für die vorher umfangreiches Regelwissen akquiriert wurde.

Die Implementierung des Konzeptes dient der Verarbeitung von Bauplänen, die aus orthogonalen dreidimensionalen Objekten bestehen, welche folglich Gegenstand der zur Erklärung benutzten Beispiele sind. Getestet wurde System durch Verarbeitung komplexer Layouts mit bis zu mehreren Tausend verschiedenen Objekten aus den Bereichen Konstruktion, Installationsplanung und Inneneinrichtung. Die Arbeit wird jedoch zum einen zeigen, daß das System nicht nur Baupläne sämtlicher Teilbereiche des Bauwesen, sondern ohne Veränderung auch Layouts anderer Anwendungsbereiche verarbeiten kann. Zum anderen wird dargelegt, was verändert werden müßte,

[1] **S**trukturorientiertes **F**all**B**asiertes **S**chließen

um auch Layouts mit nicht-orthogonalen Objekten oder Strukturen beliebiger anderer Anwendungsgebiete zu verarbeiten. Die breite Einsetzbarkeit des Konzeptes liegt an der Allgemeingültigkeit der einzigen verwendeten Heuristik: *Probleme gleicher Struktur können auf die gleiche Weise gelöst werden.*

Die zur Übertragung auf alternative Anwendungsgebiete benötigten Veränderungen betreffen ausschließlich das anwendungsspezifische Wissen, dessen Inhalt, Benutzung und Auswirkung in der Arbeit detailliert beschrieben wird. Außerhalb der anwendungsspezifischen Wissensbasis ist keine Veränderung notwendig, weswegen ich das Konzept als generisch bezeichne. Die einzige Beschränkung der Anwendbarkeit des Konzeptes liegt in der geforderten Struktur, bestehend aus Objekten und Relationen. Die zu bearbeitenden Probleme und Lösungen müssen entweder Objekte und Beziehungen zwischen Objekten enthalten, oder die Beziehungen müssen erkenn- und rekonstruierbar sein. Für derartige strukturierte Anwendungsbereiche finden sich in der Arbeit viele Beispiele.

1.1 Motivation

Das zur Erstellung eines Bauplans benötigte Wissen ist extrem komplex, auf Experten mehrerer, sich gegenseitig beeinflussender Teilbereiche verteilt und teilweise unvollständig und widersprüchlich. Daher ist es unmöglich, ein Regelsystem zu erstellen, das einen Bauplan unter allen Teilaspekten prüft, korrigiert, verfeinert oder ihn ergänzt. Ausschließlich für Teilbereiche können Regelsysteme erstellt werden, die sich in der Praxis hauptsächlich mit statischen oder technischen Problemen beschäftigen. Im Gegensatz dazu liegen in den Architektur- und Ingenieurbüros Beispiele für Lösungen von Problemen vieler Teilbereiche in Form von Bauplänen vor. Diese fertigen Baupläne reflektieren insbesondere auch den Stil, also die für jedes Büro unterschiedlich gewichteten Ansprüche an die verschiedenen Teilaspekte eines Bauobjektes. Sie repräsentieren die Erfahrung und Charakteristika eines Büros. So bevorzugt zum Beispiel ein Büro dreieckige oder runde Fenster in den Giebeln und hat gute Erfahrung mit den Fenstern eines bestimmten Herstellers gesammelt. Macht man die entsprechenden fertigen Baupläne für den Computer nutzbar, kann der Computer in allen Teilbereichen, für die bereits erfaßte Pläne vorliegen, konstruktive und dem einzelnen Büro angepaßte Unterstützung bieten. Unter diesem Ziel stand die Entwicklung des in dieser Arbeit beschriebenen Konzeptes.

1.2 Aufgabenstellung und Lösung

Die Aufgabenstellung dieser Arbeit bestand darin, ein System zu konzipieren und zu realisieren, das die automatische Anpassung komplexer Layouts auf ein gegebenes neues Layout erlaubt, ohne vorher lange Zeit Wissen in dem Anwendungsbereich zu akquirieren und zu modellieren. Entstanden sind ein Prototyp und ein Konzept, welche nicht nur zur Verarbeitung von Layouts, sondern auch für viele andere Anwendungsbereiche benutzbar ist. Der Verzicht auf umfangreiches anwendungsspezifisches Wissen macht die Übertragung auf andere Anwendungsgebiete zusätzlich attraktiv.

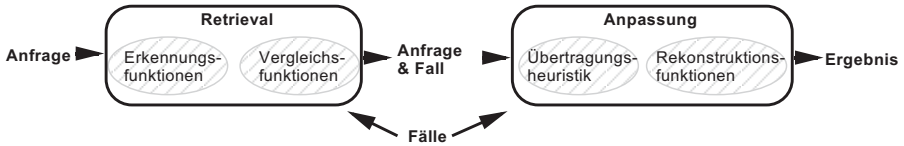

Abbildung 1.1: Diese Abbildung zeigt die vier „Löcher", die es im vorgeschlagenen Konzept mit Regelwissen zu füllen gilt.

Im fallbasierten Schließen wird die durch einen Benutzer gegebene Aufgabenstellung als „Anfrage" bezeichnet. Das „Retrieval" sucht eine zur Anfrage passende Problemlösung, welche „Fall" genannt wird (vergleiche Abbildung 1.1). Aus dem gefundenen Fall überträgt die Anpassung Information, die gegebenenfalls an die Anfrage angepaßt wird.

Der Kern des Konzeptes ist die Identifikation und Verarbeitung des Wissens, das zur fallbasierten Bearbeitung eines Anwendungsbereiches benötigt wird. Abbildung 1.1 zeigt vier „Löcher", die es im vorgeschlagenen Konzept mit Wissen zu füllen gilt. Die Erkennungs- und Rekonstruktionsfunktion erkennen und rekonstruieren Abhängigkeiten, die in einer Anfrage und einem Fall gelten. Auf diese Weise erschliessen sie Wissen, welches nur implizit einem Fall vorhanden ist. Gegeben eine Anfrage, wird anschließend mit Hilfe der Vergleichsfunktionen ein geeigneter Fall in der Menge bekannter Fälle gesucht. Aus diesem wählt die Übertragungsheuristik diejenigen Teile aus, die in der Anfrage rekonstruiert werden sollen. Da es für die Vergleichsfunktionen und die Übertragungsheuristik generisches Wissen gibt, wird nur für die Erkennungs- und Rekonstruktionsfunktionen anwendungsabhängige Modellierung zwingend benötigt. Diese Modellierung ist jedoch nachweisbar einfacher, als eine vergleichbare Modellierung zum regelbasierten Schließen.

Am Beispiel der Verarbeitung orthogonaler Layouts[2] aus dem Bereich des Bauwesens zeigt die Arbeit, wie einfache räumliche Relationen zum strukturorientierten fallbasierten Schließen benutzt werden können, da sie implizit komplexe Relationen beinhalten. Alle vorgestellten Konzepte werden jedoch auch generisch und für andere Anwendungsgebiete beschrieben.

1.3 Gliederung

Um dem Leser den Zugang zu den einzelnen Kapiteln zu erleichtern, beginnt jedes mit einer Sammlung von Fragen, die in dem Kapitel bearbeitet werden, und schließt mit einer Zusammenfassung der Antworten. Einen Überblick über die Arbeit als Ganzes geben die folgenden Absätze, wobei auf ausgewählte Fragestellungen Bezug genommen wird.

Die ganze Arbeit steht unter der Maxime, mit möglichst wenig und genau spezifiziertem anwendungsspezifischem Wissen möglichst weit zu kommen. Deshalb wird **Kapitel 2** direkt zu Beginn zeigen, welches Wissen zum „Stopfen der Löcher" benutzt wird, und wieso zum Beispiel durch einfache räumliche Relationen komplexe Zusammenhänge übertragen werden können.

Das in dieser Arbeit entwickelte Konzept kann nur deshalb generisch sein, weil die verschiedenen Teilschritte allgemeingültige Heuristiken verwenden, aus den vorhandenen anwendungsspezifischen Fällen Wissen ableiten und der Benutzer jederzeit steuernd eingreifen kann. Unter Bezug auf das klassische fallbasierte Schließen werden in **Kapitel 3** der Rahmen des SFBS und die benutzten Heuristiken beschrieben. Eine wesentliche steuernde Benutzeraktion ist zum Beispiel die Entscheidung, ob eine Anfrage korrigiert oder ergänzt werden soll.

Kapitel 4 beschäftigt sich mit dem theoretisch interessantesten Teil der Arbeit, dem Vergleich der Struktur einer Anfrage mit der eines Falles. Der Vergleich dient dabei der Suche nach einem zur Anfrage passenden Fall und der Identifikation gemeinsamer Teilstrukturen. Für diese Aufgabe wird ein Graphmatching-Algorithmus von [Barrow und Burstall, 1976] vorgestellt, dessen Effizienz wesentlich gesteigert, zur Lösung mehrere Teilprobleme verfeinert und dann zur Lösung der Retrieval-Aufgabe doch weitgehend ersetzt. Zur Anpassung wird er jedoch weiterhin essentiell benötigt.

Kapitels 5 präsentiert hauptsächlich den Anpassungsalgorithmus, der mit Hilfe der Rekonstruktionsfunktionen und aufbauend auf dem Ergebnis

[2] Orthogonale Layouts bestehen aus Objekten orthogonaler Geometrie in einem orthogonal ausgerichteten Raum.

des Strukturvergleiches die Anpassungsaufgabe löst. Am Beispiel der Verarbeitung von Bauplänen wird gezeigt, wie durch Kombination der Anpassung von Struktur und Objekteigenschaften viele der auftretenden Konflikte gelöst werden können.

Kapitel 6 beschreibt die Umsetzung des Konzeptes zur Bearbeitung von Bauplänen. Da die modellierten Relationen im wesentlichen räumliche Topologien beschreiben, heißt das entstandene Werkzeug TOPO. Anhand einer konkreten Problemlösung zeigt das Kapitel die Funktionalität und mögliche Benutzerinteraktionen und schließt mit einer Bewertung des Laufzeitverhaltens und der erzielten Lösungsqualität. TOPO ist im Rahmen des Projektes FABEL[3]entstanden. Im Projekt FABEL beschäftigten sich insgesamt fast 20 Mitarbeiter vier Jahre lang mit allen Aspekten des fallbasierten Schließens rund um die Unterstützung des Entwurfs von Bauplänen.

Um die Übertragbarkeit des Konzeptes zu zeigen, beschreibt **Kapitel 7** die Anwendung des Konzeptes auf verschiedene Bereiche, seine Verfeinerung für Teilaufgaben des Layouts und eine Verallgemeinerung in Form eines generischen Werkzeuges zum SFBS.

Kapitel 8 beschreibt andere Ansätze unter drei Aspekten. Zum einen werden interessante Strukturvergleiche vorgestellt und deren fehlende allgemeine Anwendbarkeit diskutiert. Zum zweiten gibt es einen Ansatz aus dem Bereich der Chemie, der dem gleichen Algorithmus zum Vergleich zweier Strukturen basiert, nur in einer anderen Anwendung. Zum dritten werden einige weitere bekannte spezialisierte Ansätze des fallbasierten Schließen beschrieben und mit dem Konzept des SFBS verglichen, um den Anspruch des SFBS auf allgemeine Anwendbarkeit zu unterstützen.

Im abschließenden **Kapitel 9** werden noch einmal die wichtigsten Bestandteile des vorgestellten Konzeptes zusammengefaßt und in den bisherigen Kapiteln verankert. Das Ende des Kapitels gibt einen Überblick über die in der Arbeit diskutierten möglichen Erweiterungen einiger Teilaspekte.

[3]Das Verbundvorhaben FABEL wurde vom Bundesminister für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Kennzeichen „01IW 104“ gefördert. Die Projektpartner waren die GMD – Forschungszentrum Informationstechnik GmbH, Sankt Augustin, BSR Consulting GmbH, München, die Technische Universität Dresden, HWTK Leipzig, die Universität Freiburg und die Universität Karlsruhe.

Kapitel 2

Wissensmodellierung für orthogonale Layouts

Folgende Fragen werden in diesem Kapitel beantwortet:

1. Welches Wissen benötigt das SFBS wozu?
2. Wie sieht das benötigte Wissen am Beispiel der Verarbeitung orthogonaler Layouts aus?
3. Wieso sind die Erkennungsfunktionen so wichtig?
4. Wieso reichen primitive räumliche Relationen zur Repräsentation und Verarbeitung komplexer Sachverhalte aus?
5. Wieso braucht man zum SFBS weniger Wissen als für einen rein regelbasierten Ansatz?

Dieses Kapitel beschreibt am Beispiel der Erstellung orthogonaler Layouts das für das Konzept des strukturorientierten fallbasierten Schließens (SFBS) benötigte Modell. Orthogonale Layouts bestehen aus dreidimensionalen Objekten, deren Geometrie durch Quader beschrieben werden, welche parallel zu den Dimensionsachsen ausgerichtet sind. Die Erstellung orthogonaler Layouts bietet sich als Beispiel aus drei Gründen an. Zum einen sind die Zusammenhänge intuitiv verständlich. Zum anderen ist das Anwendungsgebiet sehr allgemein und zum dritten wurde das SFBS für dieses Anwendungsgebiet im Rahmen des Projektes FABEL und dieser Arbeit implementiert.

Das folgende Modell, und damit auch das vorgestellte Konzept, deckt ein breites Anwendungsspektrum ab, ist gleichzeitig aber detailliert genug,

um nützliche Unterstützung zu bieten. Trotzdem läßt sich das Modell noch weiter in der einen Richtung generalisieren und in anderen Richtungen spezialisieren. Diese Möglichkeiten werden, neben dem in diesem Kapitel behandelten Modell für orthogonale Layouts, in Abbildung 2.1 vorgestellt und in Kapitel 7 vertieft.

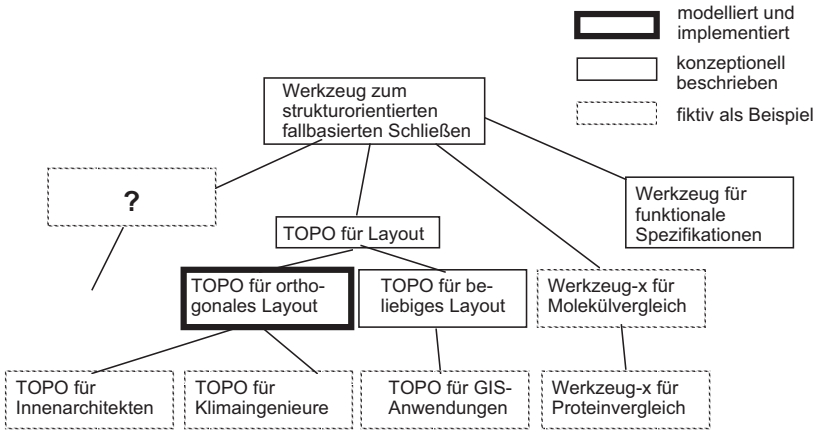

Abbildung 2.1: Abgrenzung der in diesem Kapitel behandelten Modellierung.

TOPO unterstützt die Konstruktion orthogonaler Layouts. Sowohl die Anfrage als auch das Ergebnis und die dynamische Fallbasis bestehen ausschließlich aus Layouts. Abbildung 2.2 zeigt ein Schema des benötigten Wissens.

Die Fallbasis enthält die Menge der bekannten Layouts, die im folgenden Layoutfälle genannt werden. Die Fallbasis kann dynamisch vom Benutzer verändert und ergänzt werden. Abschnitt 2.1 beschreibt die Bestandteile eines Layouts. An die Layoutfälle stellt TOPO keinerlei Anforderungen, weder bezüglich des Inhalts noch der Qualität. Das Ergebnis wird jedoch desto besser sein, je dichter der Inhalt den Bereich der möglichen Anfragen abdeckt und je besser die Qualität der Layoutfälle ist. Dieser Zusammenhang wird in Abschnitt 7.1 detailliert diskutiert.

Die statische Wissensbasis (vergleiche Abbildung 2.2) ermöglicht das Erkennen, den Vergleich und die Rekonstruktion von dreidimensionalen orthogonalen Relationen und stellt eine Heuristik zur Verfügung, um zu Entscheiden, welche Objekte und Relationen aus der Fallbasis auf das Anfragelayout

übertragen werden.

Das Ergebnis besteht aus dem Anfragelayout, auf das Objekte und Relationen aus den aus Layoutfällen übertragen wurden. Um die in Layouts implizit enthaltene Struktur zu erkennen, zwei Strukturen zu vergleichen und Teile zu übertragen, wird zusätzliches Wissen benötigt. Dieses Wissen ist Bestandteil der statischen Wissensbasis.

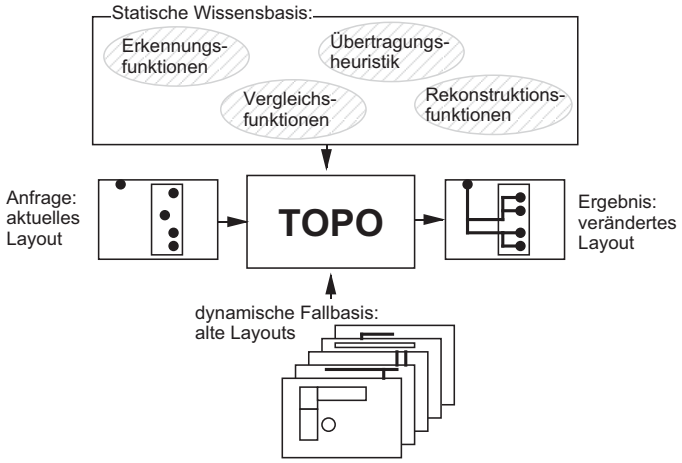

Abbildung 2.2: Schematische Darstellung des von TOPO benötigten Wissens.

Die folgenden Abschnitte motivieren und beschreiben die Interpretation von Layouts und begründen den Vorteil der Wiederverwendung interpretierter Layouts gegenüber einem rein regelbasiertem System.

2.1 Explizite Layoutbestandteile

Elektronisch verfügbare Layouts sind meistens mit Werkzeugen des computerunterstützen Designs (CAD) erstellt worden, die zum Teil auf die Arbeit von Architekten (CAAD) und anderen Berufen spezialisiert sind. So erstellte Layouts enthalten zum Großteil die Beschreibung geometrischer Objekte. Im orthogonalen Layout wird die Geometrie der Objekte durch achsenparallele dreidimensionale Quader approximiert. Diese Objekte verfügen je nach benutztem Werkzeug und Arbeitsweise des Anwenders über unterschiedliche Eigenschaften. Besitzen die Objekte neben ihrer Geometrie keine

weiteren semantischen Eigenschaften, so kann natürlich nur diese Information zum Schließen verwendet werden. In vielen Anwendungsbereichen, wie zum Beispiel der Architektur, Maschinenbau, Anlagenplanung oder auch dem Bereich der geographischen Informationssysteme gibt es weitere relevante objektspezifische Eigenschaften, die sich zur Unterstützung des Benutzers ausnutzen lassen. Beispiele sind Materialart, Kosten, Funktionalität, Bestellnummer, Bebauungsart und vieles mehr. Bei komponentenorientierten CAD-Werkzeugen ist es möglich, Komponentenbibliotheken zu erstellen und zu benutzten. Erstellt der Benutzer ein Layout nun unter Verwendung der Komponenten, so enthält es neben rein geometrischer Information auch die Beschreibung der Eigenschaften der Komponenten. Die in dieser Arbeit zur Erläuterung des Textes verwendeten Beispiele benutzen Objekte, deren Eigenschaften sie als Stühle und Tische oder als Auslaßöffnungen, Verbindungsleitungen und Stammleitungen des Klimasystems kennzeichnen. Zum in Kapitel 6.6 beschriebenen Test des Systems wurden komplexe Layouts mit bis zu mehreren Tausend verschiedener Objekte aus den Bereichen Konstruktion, Installationsplanung und Inneneinrichtung verwendet. Abbildung 2.3 zeigt ein Layout durchschnittlicher Größe, welches die Erschließung einer Geschoßhälfte mit unterschiedlich abstrakten Objekten des Zuluftsystems enthält. Außer der Geometrie und den Objekteigenschaften enthält es keine weitere Information, insbesondere Relationen. Komplexe Berechnungsmodelle, welche zum Beispiel zusätzlich an statische Pläne angefügt werden, sind so anwendungsspezifisch, daß sie im Rahmen des hier vorgestellten allgemeinen Konzeptes nicht beschrieben werden.

2.2 Implizite Layoutbestandteile

Bisherige Ansätze verarbeiten hauptsächlich die bisher beschriebenen expliziten Bestandteile. Neben den expliziten Layoutbestandteilen enthalten Layouts jedoch weitere Informationen. Ein Bauingenieur sieht zum Beispiel, daß zwei Leitungen „verbunden" sind, eine Stütze einen Träger „trägt", genauso wie ein Innenarchitekt sieht, daß zwei Möbel aufeinander „farblich abgestimmt" sind. Diese Beziehungen zwischen einzelnen Objekten und Objektmengen sind meistens nur implizit vorhanden, werden jedoch von einem Experten direkt erkannt. Die automatische Erkennung eines Teiles dieser Relationen ist der Schlüssel zum sinnvollen Vergleichen und Übertragen von Fällen und ist somit einer der wichtigsten Bestandteile des SFBS. Zur technischen Verarbeitung betrachtet das SFBS ein Layout als einen Graph, dessen

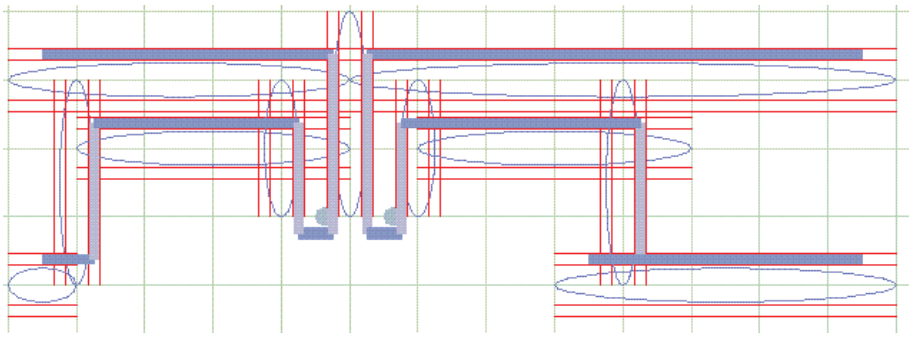

Abbildung 2.3: Ein typisches Ausschnitt eines orthogonalen Bauplans, bestehend aus unterschiedlich abstrakten Objekten des Zuluftsystems.

Knoten die Objekte und dessen Kanten die Beziehungen repräsentieren. Im weiteren wird dieser Graph als „Strukturgraph" oder einfacher, als „Struktur" eines Layouts bezeichnet.

Wären dem Unterstützungssystem, zusätzlich zu den Objekten mit ihren Eigenschaften, alle Beziehungen bekannt, so würde es wesentliche Teile des Vokabulars kennen, mit dem ein Experte ein von ihm erstelltes Layout beschreiben könnte. Darauf aufbauend könnten Layouts im Sinne des Experten verglichen und über ihnen geschlossen werden. In manchen Anwendungsgebieten, wie zum Beispiel bei Netzplänen, Projektplänen oder der Verarbeitung von Molekülstrukturen sind die Beziehungen üblicherweise explizit in einem Plan enthalten. Im CAD-Layout ist es jedoch für einen Experten unnatürlich, diese Beziehungen explizit anzugeben, da er sie in der Zeichnung „sieht". Zum anderen wäre die Erstellung und Wartung der Beschreibung der Beziehungen für den Experten sehr aufwendig. Außerdem würde die Wartung sehr unübersichtlich, da es in kaum einem CAD, GIS (Geographic Information System) oder anderem Graphikeditor möglich ist, diese Beziehungen anzugeben und zu verwalten. Die übliche Möglichkeit, Mengen von geometrischen Objekten in Blöcken zu gruppieren, reicht aus verschiedenen Gründen nicht aus, um Beziehungen zu repräsentieren. Das entscheidende Problem ist, daß ein Objekt des Layouts nicht in mehreren Blöcken enthalten sein kann und somit seine verschiedenen Beziehungen zu verschiedenen anderen Objekten nicht repräsentiert werden können. Um trotzdem die Beziehungen zum Schließen benutzen zu können, muß das Unterstützungssystem sie „er-

kennen". Dieses „Erkennen" ist eher ein „Interpretieren", da es versucht, aus der Sicht eines Experten Zusammenhänge in ein Layout hineinzulesen.

Was am Beispiel „verbunden", „trägt", „farblich passend" noch mehr oder weniger machbar erscheint, ist in der Realität komplexer Layouts unterschiedlichster Anwendungsbereiche unrealistisch. Genauso, wie man kaum alle Regeln modellieren kann, kann man nicht alle möglichen Beziehungen definieren. Im Gegensatz zu den Regeln gibt es jedoch Beziehungen, die in allen Layouts eine wichtige Rolle spielen und auf denen viele anwendungsspezifischen Beziehungen basieren, die räumlichen Beziehungen.

Der folgende Abschnitt wird neben Beispielen für die Modellierung räumlicher Beziehungen im orthogonalen Raum auch alle weiteren für das SFBS benötigten Bestandteile des statischen Wissens am Beispiel der Anforderungen von TOPO beschreiben.

2.3 Benötigtes Wissen

Der Vergleich eines Falles mit einer Anfrage erfordert nur in strukturierten Anwendungsbereichen Wissen, wie ich es im weiteren beschreibe. Im allgemeinen genügen „einfachere" attributbasierte Vergleichsfunktionen. Abschnitt 3.2 wird sich mit dem Verhältnis zwischen attributbasierten und strukturbasierten Vergleichsfunktionen beschäftigen.

Strukturbasierte Vergleichsfunktionen vergleichen die Bestandteile einer Struktur, also Objekte und Beziehungen. Um eine Beziehung aus einem Layoutfall auf ein neues zu übertragen, zum Beispiel ein Objekt „genau so" im neuen Layout zu plazieren wie im Layoutfall, muß das Unterstützungssystem eine Beziehung nicht nur erkennen, sondern sie auch rekonstruieren können. Damit nur „sinnvolle" Übertragungen durchgeführt werden, kann man außerdem Wissen darüber bereitstellen, wann welche Übertragung „sinnvoll" ist.

Abbildung 2.4 zeigt ein Beispiel für die Benutzung der verschiedenen Wissensarten. Mit Hilfe der Erkennungs- und Vergleichsfunktionen ordnet TOPO die Zuluftleitungen aus Anfrage und Fall einander zu. Eine Übertragungsheuristik könnte feststellen, daß nur die zusätzlichen Zuluftleitungen und nicht die Möbel aus dem Fall in die Anfrage übertragen werden sollen. Anschließend wird die exakte absolute Position der übertragenen Zuluftobjekte durch die Rekonstruktionsfunktionen rekonstruiert.

Zur Übertragung kann man sich zusammen mit dem Experten viele verschiedene Heuristiken überlegen, wann welche Teile eines Falles auf eine Anfrage übertragen werden sollen. Abschnitt 5.1 stellt eine Idee vor, wie Über-

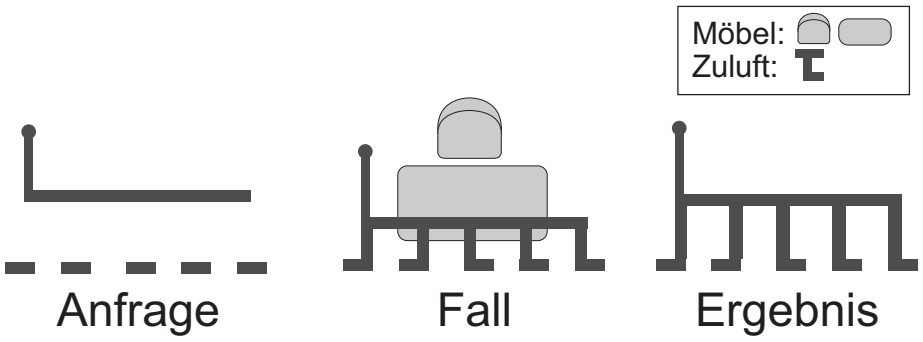

Abbildung 2.4: Ein Beispiel für eine fallbasierte Ergänzung.

tragungsheuristiken gelernt werden können. Zuvor wird dieser Abschnitt jedoch eine möglichst einfache Wissensmodellierung beschreiben, auf deren Basis die folgenden Kapitel zeigen, wie man trotz einfacher Modellierung ein breit einsetzbares Unterstützungssystem entwirft und umsetzt.

Um das SFBS in anderen Bereichen einzusetzen, muß es teilweise verändert, verfeinert oder ersetzt werden. Man kann das modellierte Wissen somit als Parameter eines generischen Werkzeuges zum SFBS betrachten, wie ich es in Kapitel 7 beschreibe.

2.3.1 Erkennung und Rekonstruktion primitiver orthogonaler räumlicher Relationen

Räumliche Relationen können die Topologie, die Richtung und die Distanz erfassen. Viele bekannte Konzepte, um räumliche Relationen zu beschreiben, basieren auf den in [Allen, 1983] beschriebenen temporalen Relationen. Einige von ihnen sind in [Guesgen und Hertzberg, 1992] beschrieben. Da die Konzepte jedoch auf zwei Dimensionen beschränkt sind und insbesondere von geometrischen Distanzen abstrahieren, wird in dieser Arbeit keines der bekannten Konzepte benutzt.

Für orthogonales Layout verwende ich alle Relationen von Allen für jeweils eine Dimension, erfasse jedoch auch die Richtung einer Beziehung und die Distanz zwischen Objekten, da alle drei Eigenschaften (Topologie, Richtung, Distanz) im Layout wichtig sind. Weil die Ausrichtung der Objekte und Relationen im dreidimensionalen orthogonalen Raum auf die drei

Dimensionsrichtungen beschränkt ist, kann man die Erkennung der dreidimensionalen Beziehungen auf die Erkennung eindimensionaler Relationen zurückführen, wodurch es für orthogonale Objekte prinzipiell die gleichen Beziehungen gibt, wie für Zeitintervalle (vergleiche Abbildung 2.5). Man projeziert die Objekte eines Layouts auf die drei Koordinatenachsen und bestimmt je Dimension zwischen den so erhaltenen Intervallen eine Relation, die die Topologie und Distanz ausgedrückt. Der Aspekt der Richtung ist durch die Unterscheidung der drei Dimensionen repräsentiert.

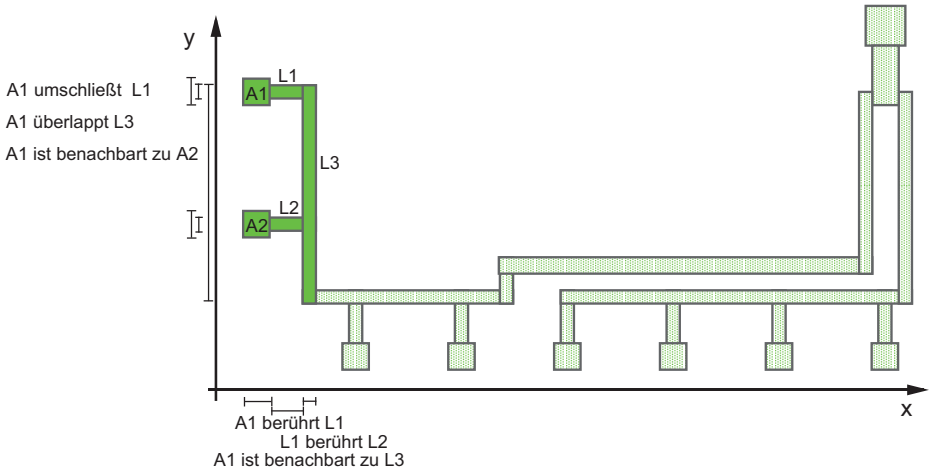

Abbildung 2.5: Um 3-dimensionale Relationen aus einer Kombination dreier eindimensionaler Relationen zwischen Intervallen zu bilden, werden alle Objekte eines Layouts auf die orthogonalen Ebenen entlang der Koordinatenachsen projeziert.

Die in Abbildung 2.6 gezeigten acht eindimensionalen Relationen beschreiben die Beziehungen zwischen Intervallen. Damit nur „benachbarte" Objekte miteinander in Beziehung stehen, werden nur Objektpaare betrachtet, die weniger als ein festzusetzendes Vielfaches N der maximalen Objektgröße der beteiligten Objekte voneinander entfernt sind. Damit auch sehr kleine Objekte miteinander in Beziehung gesetzt werden können, werden zusätzlich Objekte betrachtet, die weniger als eine Maximaldistanz D auseinander sind. Das Vielfache N und die Maximaldistanz D sind anwendungsabhängig vorzugeben. Setzt man zum Beispiel N auf 2, so stehen zwei Räume mit 10m Durchmesser und einer Distanz von 15m noch in Beziehung

zueinander, nicht aber zwei 15m voneinander entfernte Stühle. Ein Beispiel für sehr kleine Objekte der Gebäudeplanung sind Auslaßöffnungen der Abluft. „Benachbarte" Auslaßöffnungen sind selten weniger als die doppelte Objektgröße voneinander entfernt. Um trotzdem Beziehungen zwischen ihnen zu erkennen, ist in den Beispielen D auf 4m gesetzt. Die Auswirkungen möglicher Werte für D und N werden in Kapitel 6.6 diskutiert.

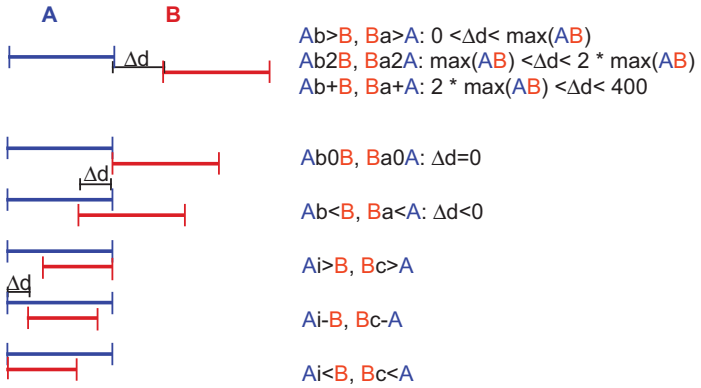

Abbildung 2.6: Acht verschiedene eindimensionale Relationen zwischen Objekten A und B, gegebenenfalls mit Distanzparameter Δd.

Wenn sich zwei Objekte A und B berühren, ist die Distanz Δd gleich Null und implizit in der topologischen Relation enthalten. Wenn sie sich nicht berühren, so unterscheide ich für die unterschiedlichen Entfernungen Δd drei Relationstypen in Abhängigkeit von der Größe der beteiligten Objekte (max(AB)) (vergleiche Abbildung 2.6). Sie basieren auf der anwendungsspezifischen Festsetzung des Vielfachen N auf 2 und der Maximaldistanz D auf 4m. Falls vorhanden, wird die relative Entfernung $\Delta d/max(AB)$ bei der Erkennung in der Relation gespeichert. Auf diese Weise ist eine eindeutige Rekonstruktion für alle acht eindimensionalen Relationen möglich und somit auch die Rekonstruktion jeder aus einer Kombination von eindimensionalen Relationen bestehenden dreidimensionalen Relation.

Das Zusammenspiel von Erkennungs- und Rekonstruktionsfunktionen ist für das SFBS unbedingt erforderlich, da es Voraussetzung für die Anpassung ist. Die Erkennungs- und Rekonstruktionsfunktion einer Relation definieren zusammen ein Constraint, welches sich selbst in Fällen instanziieren und in Anfragen erfüllen kann.

Da die acht Relationen auch die Richtung beschreiben, es also wichtig ist, ob Objekt A in Relation R zu B steht oder umgekehrt (zum Beispiel: A vor B $\neq$ B vor A), kann ein Objektpaar in einer von 16 ($2 * 8$, vergleiche Abbildung 2.6) möglichen Relationen je Dimension stehen. Setzt man die eindimensionalen Beziehungen zweier Objekte zu einer dreidimensionalen zusammen, so gibt es $16^3 = 4096$ verschiedene Relationen. Theoretisch denkbar wären zusätzliche Beziehungen, die ausdrücken, daß zwei Objekte in keiner Beziehung zueinander stehen. Dies gilt für alle Objekte, die in mindestens einer Dimensionen sowohl mehr als das Vielfache N der maximalen Objektgröße der beteiligten Objekte, als auch mehr als die Maximaldistanz D voneinander entfernt sind. Diese Beziehungen werden jedoch von TOPO nicht benutzt und brauchen deshalb nicht explizit repräsentiert zu werden. Implizit sind sie durch das Fehlen jeglicher Beziehung zwischen zwei Objekten erkennbar.

Die Erkennung und Rekonstruktion der beschriebenen Relationen ist für orthogonales Layout eindeutig und für die Beziehungen zwischen „benachbarten" Objekten vollständig. Auch Drehungen von Teilen eines Layouts um Vielfache von 90° und Spiegelungen führen zu eindeutigen Transformationen der Relationen, so daß auch die Strukturen derart transformierter Layouts miteinander verglichen werden können. Abschnitt 4.2.1 behandelt dieses Teilproblem des Vergleichs. Im nicht-orthogonalen Layout ist es wesentlich schwieriger, alle möglichen Beziehungen eindeutig zu repräsentieren. Abschnitt 7.4.2 wird diese Schwierigkeiten diskutieren und einen möglichen Lösungsweg zeigen.

2.3.2 Vergleichsfunktionen für Objekte und Relationen

Die Vergleichsfunktionen sollen prüfen, ob zwei Objekte oder Relationen aus zwei Layouts einander zugeordnet werden können. Bei der Implementierung von TOPO habe ich mich für Vergleichsfunktionen entschieden, die auf {0,1} abbilden und nicht auf das Intervall [0,1], wie es typisch für Ähnlichkeitsfunktionen ist. Deswegen spreche ich im folgenden davon, daß zwei Objekte oder Relationen *kompatibel* sind, wenn ihre Ähnlichkeit gleich eins ist. Dies bedeutet, daß sie während eines Strukturvergleiches einander zugeordnet werden dürfen. Die positiven und negativen Konsequenzen der Möglichkeit, die Ähnlichkeit von Objekten und Relationen auf das Intervall [0,1] abzubilden, wird bezüglich der Wissensmodellierung in Abschnitt 7.3 und bezüglich der Konsequenzen für den Graphvergleich in Abschnitt 4.2.3 diskutiert.

Um die Ähnlichkeit zweier Objekte zu bestimmen, ist anwendungsspezi-

fisches Wissen nötig. Um möglichst allgemein zu bleiben, habe ich mich bei der Modellierung für TOPO darauf beschränkt, daß zwei Objekte kompatibel sind, wenn ihre semantischen Eigenschaften identisch sind.

Zwei dreidimensionale Relationen sind für TOPO kompatibel, wenn ihre drei eindimensionalen Relationen bis auf Δd identisch sind und die beteiligten Objekte kompatibel. Genauso wie für die Objekte ist es auch möglich, für die Relationen kompliziertere Ähnlichkeitsfunktion zu definieren, wie zum Beispiel der Ansatz des Conceptual Neighbourhoods [Freksa, 1992a]. Auch dieser Ansatz wird in Kapitel 4.2.3 vorgestellt werden.

2.3.3 Übertragung sinnvoller Zusammenhänge

Es ist sinnvoll, eine Relation aus einem Layoutfall auf ein Anfragelayout zu übertragen, wenn sie zur Lösung eines im Anfragelayout enthaltenen Problems gehört. Leider ist bei einem Layout nicht ohne zusätzliches Wissen zu erkennen, was das Problem und was die dazugehörige Lösung ist. Deswegen kann man nicht entscheiden, was „sinnvoller Weise" aus einem Layoutfall auf ein Anfragelayout übertragen werden kann. Im allgemeinen können alle diejenigen Relationen aus einem Layoutfall übertragen werden, die Fallobjekte betreffen, welche Objekten aus dem Anfragelayout zugeordnet wurden. Werden dabei Fallobjekte in das Anfragelayout übertragen, so muß man versuchen, auch deren Relationen mit zu übertragen. Dies kann zu einer iterativen Übertragung von Fallobjekten in das Anfragelayout führen. Diese allgemeine Heuristik kann zum einen auch durch anwendungsspezifisches Wissen ergänzt werden (vergleiche Abschnitt 7.3). Zum anderen kann eine Verfeinerung der Übertragungsheuristik automatisch gelernt werden. Abschnitt 5.1 wird Ideen dazu präsentieren. Dieses zusätzliche Wissen ist aber für den Einsatz des Konzeptes nicht notwendig und wird hier nicht weiter besprochen.

2.4 Vorteile des strukturorientierten fallbasierten Konzeptes

Die Kombination der Modellierung räumlicher Beziehungen und der Nutzung von Layoutfällen ergibt Wissen, dessen regelbasierte Modellierung aufwendig oder gar unmöglich ist. Zur Verdeutlichung betrachte man folgende Situation: Das Anfragelayout beinhalte ein Büro, das einen Schreibtisch, aber noch keinen Schreibtischstuhl enthält.

Um es wissensbasiert zu ergänzen, sind Regeln wie die folgenden nötig:

1. Vor jeden Schreibtisch gehört ein Schreibtischstuhl.

2. Die Farbe, das Material und die Preisklasse der Möbel eines Raumes sollten zueinander passend gewählt werden.

Um diese Regeln so zu modellieren, daß ein Unterstützungssystem sie benutzen kann, erfordert es viele zusätzliche Details:

Die Konstruktionsfunktion „vor" (um die Position des Stuhles in Abhängigkeit von der Position des Schreibtisches zu bestimmen), die Zugehörigkeit von Schreibtisch und Stuhl zur Gruppe der Möbel und die Definition von zueinander passenden Farben, Materialien und Preisklassen. Diese Definitionen sind zum Teil von den Objekten und ihren Eigenschaften abhängig, und es gibt fast immer Ausnahmen: Vor einen Schreibtisch aus Eiche paßt kein Stuhl aus Acryl, ist der Tisch jedoch schwarz lackiert, würde der Stuhl schon eher passen. Eine weitere Schwierigkeit ist, daß diese Definition zum Teil subjektiv und damit benutzerabhängig sind. Man müßte also für jeden Benutzer eigene Regeln definieren oder sie zumindest benutzerabhängig parametrisieren. Ist ein entsprechender Stuhl bestimmt, kann es immer noch sein, daß ein Stuhl mit den geforderten Eigenschaften gar nicht existiert und somit die Regeln keine Lösung zulassen. Entsprechend sind weitere Regeln zur Relaxation nötig. Sind selbst diese modelliert, so ist immer noch nicht sichergestellt, daß die Regeln vollständig sind.

Man sieht, daß selbst dieses kleine Beispiel einen hohen Modellierungsaufwand für ein regelbasiertes System erfordert. Geht es nun nicht nur um Schreibtische und Schreibtischstühle, sondern zum Beispiel um das Layout ganzer Gebäude, so kann bisher kein regelbasiertes System ein unter allen denkbaren Gesichtspunkten „beste" Lösung finden. Natürlich kann dies auch kein anderes System und es ist in vielen komplexen Anwendungen nicht definiert, was die „beste" Lösung ist. Die Aussage soll jedoch zeigen, daß es in einigen komplexen Anwendungen keinen Sinn macht, einem System vorzuwerfen, daß es nicht die „beste" Lösung findet. Statt dessen sollte ein System mit Hilfe des vorhandenen Wissens eine möglichst gute Lösung finden. Das im folgenden beschriebene fallbasierte Konzept versucht, möglichst gute Lösungen zu finden, die häufig sogar über weitere positive Eigenschaften verfügen, die durch das modellierte Wissen nicht erkennbar und schon gar nicht konstruierbar sind.

Geht man fallbasiert vor und benutzt neben einer geeigneten Fallbasis [1] die in Abschnitt 2.3 beschriebenen räumlichen Beziehungen und Layoutfälle zur Ergänzung des Schreibtischstuhles, so reicht folgende breit einsetzbare Heuristik aus:

1. Gilt eine Beziehung zwischen zwei Objekten A und B in einem zum Anfragelayout „ähnlichen" Layoutfall, so soll sie in dem Anfragelayout rekonstruiert werden, falls eines von beiden Objekten in der Anfrage enthalten ist.

Bei der Anwendung dieser Regel entstehen neue Objekte und Relationen in der Anfrage. Durch Iteration werden dabei auch Objekte erzeugt, die in Beziehung zu im vorhergehenden Iterationsschritt gerade erst erzeugten Objekten stehen.

Das Ergebnis der Übertragung von Teilen von Layoutfällen wird selten die unter allen denkbaren Gesichtspunkten „beste" Lösung ergeben. Es werden jedoch viele Beziehungen aus dem Layoutfall übertragen werden, die noch nicht einmal explizit modelliert oder erkannt sind. Ist im Beispiel mithilfe der Ähnlichkeitsfunktion ein Layoutfall ausgewählt worden, dessen Schreibtisch zu dem des Anfragelayouts ähnlich oder identisch ist, so wird der Stuhl des Layoutfalles auch zum aktuellen Schreibtisch in einigen Aspekten „passen".

Falls man sicherstellen will, daß zum Beispiel der Farbaspekt beim Vergleich und der Übertragung berücksichtigt wird, kann man ihn zusätzlich zu den räumlichen Beziehungen modellieren, wie es in Abschnitt 7.3 gezeigt wird. Auf diese Weise kann man die Qualität im gleichem Maße steigern, wie man die Qualität eines regelbasierten Ergebnisses durch weitere Regelmodellierung steigern kann. Der Unterschied ist, daß bei einer Regel angegeben werden muß, für welche Objekte eine Beziehung gelten soll und welche weiteren Vorbedingungen gelten müssen. Durch die Benutzung von Layoutfällen werden diese Vorgaben ersetzt durch:

- Für welche Objekte soll eine Beziehung gelten?
  Für diejenigen, für die sie in einem Layoutfall gilt!
- Unter welchen Vorbedingungen soll sie gelten?
  Wenn der Layoutfall zum Anfragelayout ähnlich ist! Da im Layoutfall die Vorbedingungen scheinbar erfüllt waren, wird angenommen, daß sie auch im dazu ähnlichen Anfragelayout erfüllt sein sollen.

[1] Positive Eigenschaften einer Fallbasis werden in Abschnitt 7.1 diskutiert.

Zusammenfassung der in diesem Kapitel benutzten Annahmen:

1. Räumliche Relationen spielen eine wesentliche Rolle bei der Erstellung von Bauplänen.
2. Die relative Distanz $\Delta d/max(AB)$ zweier Objekte A und B ist wichtiger als deren absolute Distanz.
3. Die Rekonstruktionsfunktionen können die durch die Erkennungsfunktionen erkannten Relationen rekonstruieren.
4. Die gewählten primitiven Relationen sind relevant genug, um alle wichtigen Unterschiede zu repräsentieren.
5. Die gewählten primitiven Relationen sind spezifisch genug, um nur wichtige Unterschiede zu repräsentieren.

Insbesondere die letzten beiden Annahmen sind nicht vollständig erfüllt. Trotzdem bleibt die Funktionalität erhalten. Je stärker jedoch die Annahmen verletzt sind, desto schlechter wird das Ergebnis sein.

Zusammenfassung der in diesem Kapitel enthaltenen Antworten:

1. $\Rightarrow$ Zum SFBS werden Erkennungsfunktionen, Rekonstruktionsfunktionen, Vergleichsfunktionen und Übertragungsheuristiken benötigt.
2. $\Rightarrow$ Das zur Verarbeitung orthogonaler Layouts benötigte anwendungsspezifische Wissen besteht aus primitiven räumlichen Relationen, einem Identitätstest zum Vergleich von Objekten und Relationen und der allgemeinen Übertragungsheuristik.
3. $\Rightarrow$ Durch die Erkennungsfunktionen ist es möglich, Fälle zu interpretieren und dabei implizite Zusammenhänge zu erkennen. Durch Vergleich und Übertragung der erkannten Zusammenhänge kann somit implizit in den Fällen enthaltenes Wissen genutzt werden.
4. $\Rightarrow$ Primitive räumliche Relationen reichen aus, da sie implizit komplexe Zusammenhänge beschreiben.
5. $\Rightarrow$ Das SFBS benötigt weniger Regelwissen als ein generativer Ansatz, da die Vorbedingungen nicht für jede Relation einzeln definiert werden müssen.

Kapitel 3

Funktionalität des SFBS

Folgende Fragen werden in diesem Kapitel beantwortet:

1. Wie funktioniert fallbasiertes Schließen?
2. Was unterscheidet das SFBS von attributbasierten Verfahren?
3. Welche Heuristiken sind beim SFBS vorgegeben, welches Wissen kann diese Heuristiken ergänzen, und was kann man einstellen?
4. Welche Zielsetzungen gibt es und wie unterscheiden sie sich untereinander?
5. Wie können Konflikte entstehen und wie geht das Verfahren damit um?

Dieses Kapitel wird zeigen, wie das in Kapitel 2 beschriebene statische Wissen zum strukturorientierten fallbasierten Schließen (SFBS) benutzt werden kann. Hierbei wird deutlich, daß das Verfahren generisch und keineswegs auf orthogonales Layout beschränkt ist.

Das Paradigma des fallbasierten Schließens[1] wurde erstmals in [Schank, 1982] beschrieben, wurde unter anderen von [Kolodner, 1983; Hammond, 1989; Slade, 1991] weiterentwickelt und ist mittlerweile weltweit Gegenstand mehrerer Forschungsprojekte und Entwicklungen der Informatik. Der erste Abschnitt wird das Paradigma unter Bezug auf den zusammenfassenden Artikel von [Aamodt und Plaza, 1994] vorstellen. Den Vorteil der Repräsentation von Anfrage und Fällen durch Strukturen, statt durch Mengen von Attributen, erläutert Abschnitt 3.2. Auf den ersten beiden Abschnitten aufbauend zeigt Abschnitt 3.3 die verschiedenen Teilschritte des SFBS. Die

[1] Die in der Literatur häufig benutzte Abkürzung „CBR" steht für **C**ase **B**ased **R**easoning.

Abschnitte 3.4 und 3.5 beleuchten die wichtigsten Teilschritte des SFBS unter den zwei möglichen Zielsetzungen „Ergänzung" und „Korrektur" einer Anfrage.

3.1 Fallbasiertes Schließen

Abbildung 3.1 aus [Aamodt und Plaza, 1994] zeigt ein Schema des fallbasierten Schließens, wie es allgemein anerkannt ist. Andere ähnliche Darstellungen finden sich in [Kolodner, 1993].

Das von Aamodt gewählte Schema charakterisiert das fallbasierte Schließen als Zyklus aus Lösen von Problemen und Lernen aus der gerade eben durchgeführten Problemlösung (vergleiche Abbildung 3.1). Der Zyklus beginnt oben links mit einem neuen Fall (new case), auch *Anfrage* genannt, der typischerweise ein Problem enthält. Der folgende Prozeß gliedert sich in die Suche nach einem zur Anfrage passenden Fall (Retrieval), der Übertragung von Information aus dem Fall auf die Anfrage (Reuse), der Überarbeitung des Ergebnisses (Revise) und der abschließenden Übernahme des Ergebnisses in die Wissensbasis (Retain). Die Wissensbasis besteht aus anwendungsspezifischem Hintergrundwissen (General Knowledge) und einer Fallbasis (Previous cases).

Der Inhalt der Wissensbasis wird von Aamodt nicht näher spezifiziert, soll aber die genannten Teilschritte der Problemlösung ermöglichen und ist im allgemeinen anwendungsspezifisch.

Nach Aamodt besteht das *Retrieval* aus mehreren Teilschritten. Der Teilschritt „identify" repräsentiert die relevanten Beschreibungsmerkmale der Anfrage, „search" führt eine Suche anhand einfacher Ähnlichkeitsmaße durch und „initial match" ordnet Teile der Anfrage und des Suchergebnisses einander zu. Aus Effizienzgründen wird zwischen einfachen und komplexen Ähnlichkeitsmaßen unterschieden, denn das Besondere an einfachen Ähnlichkeitsmaßen ist ihre Schnelligkeit. Im Gegensatz zu diesen kann der „initial match" komplex sein. Die einfachen Ähnlichkeitsmaße werden zum Vergleich der Anfrage mit der gesamten Fallbasis benutzt und liefern eine Vorauswahl der ähnlichsten Fälle. Die komplexen Ähnlichkeitsmaße dienen zum vertiefenden Vergleich der Anfrage mit den Fällen der Vorauswahl. Den Abschluß des Retrievals bildet der Schritt „select", in dem durch Hintergrundwissen oder durch den Benutzer auf Basis des „initial match" ein Fall zur weiteren Verarbeitung ausgewählt wird.

Der zweite große Schritt ist *Reuse*. Er wird unterteilt in „copy" und „ad-

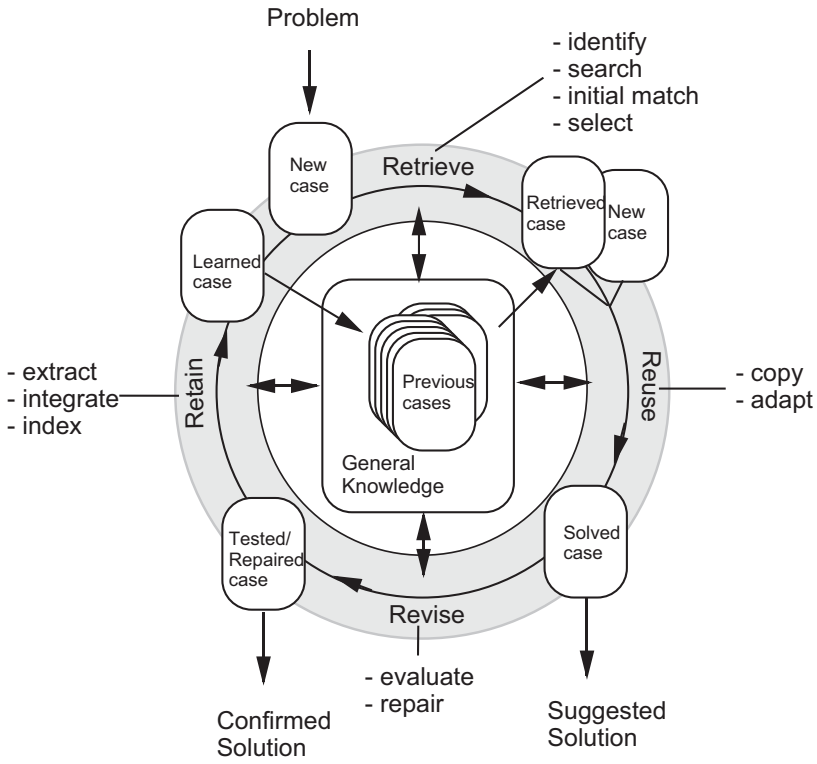

Abbildung 3.1: Schema des fallbasierten Schließens aus [Aamodt und Plaza, 1994].

apt". Der Teilschritt „copy" bestimmt die Teile des gewählten Falles, die auf die Anfrage übertragen werden sollen, und kopiert sie in die Anfrage. Die Bestandteile der Anfrage werden dabei nicht verändert. Gibt es relevante Unterschiede zwischen Fall und Anfrage, so benutzt „adapt" Transformationsregeln, die die auf die Anfrage übertragenen Teile entsprechend der relevanten Unterschiede anpaßt.

Revise bewertet („evaluate") und verbessert gegebenenfalls das bisher erzielte Ergebnis („repair"). Dazu wird ein Modell des Anwendungsbereiches benutzt oder der Benutzer gefragt.

Gute Lösungen werden im Schritt *Retain* in die Wissensbasis übernommen. Dazu werden im Teilschritt „extract" die relevanten Beschreibungsmerkmale erzeugt und auch bestimmt, welche Information in die Wissensbasis übernommen werden soll. Daraufhin wird die Wissensbasis durch „integrate" aktualisiert und „index" sortiert Index des neuen Falles in die Indexstruktur der Fallbasis ein.

Die in den bisherigen Abschnitten genannten Schritte kann man in allen fallbasierten Systemen, meist teilweise, wiederfinden. Das Retrieval ist im allgemeinen am besten verstanden, was sich auch in dessen detailierter Aufteilung widerspiegelt. Die restlichen Schritte unterscheiden sich in verschiedenen Systemen noch stark, was man in den Beiträgen zu internationalen Konferenzen zum fallbasierten Schließen[2] erkennt. Einige von diesen werden in Kapitel 8.4 diskutiert. Es gibt jedoch keinen, der, wie das hier vorgestellte Konzept, ein generisches Konzept zur Anpassung beinhaltet.

3.2 Struktur ist mehr als eine Attributmenge

Da Struktur, neben den durch Attributmengen beschriebenen Objekten, auch Beziehungen zwischen Objekten enthält[3], liefert ein Vergleich von Strukturen offensichtlich mehr Information als ein Vergleich statischer Attributmengen. Trotzdem repräsentieren viele fallbasierte Systeme Fälle und Anfragen durch Attributmengen, selbst wenn dazu von einer vorhandenen Struktur abstrahiert werden muß. Dies kann durch die abschreckende Komplexität eines Strukturvergleiches begründet sein. Die Einschränkung auf Attributmengen führt jedoch dazu, daß die Anpassung in strukturierten An-

[2] International Conference on Case Based Reasoning (ICCBR), European Workshop on Case Based Reasoning (EWCBR).

[3] Somit kann die in 2.2 definierte Struktur Beziehungen zwischen beliebigen Attributmengen repräsentieren.

wendungen nicht automatisierbar ist oder aber wiederum sehr komplex und viel anwendungsspezifisches Wissen benötigt.

Im einzelnen bietet das strukturorientierte Schließen in strukturierten Anwendungsbereichen folgende Vorteile gegenüber attributbasierten Ansätzen:

- Bei Repräsentation durch Attributmengen ist nur der Vergleich absoluter Werte, zum Beispiel absoluter Positionen möglich. Durch die Repräsentation und den Vergleich von relativen Zusammenhängen steigt die Aussagekraft des Vergleichs. Stehen zum Beispiel Stuhl und Tisch in Anfrage und Fall an völlig verschiedenen Positionen, steht jedoch beide Male der Stuhl vor dem Tisch, so wird dies nur durch den Vergleich der räumlichen Struktur erkannt.

- Ein weiterer und sehr einfach zu nutzender Vorteil des SFBS ermöglicht eine generische Bewertungsfunktion für Anfragen und Ergebnisse. Man kann die relativen Häufigkeiten der in der Fallbasis vorkommenden Relationen objektspezifisch in einer Statistik beschreiben. Als Beispiel gebe es in der Fallbasis 100 Tische, 1000 Stühle, 98-mal die Relation „Stuhl vor Tisch" und keinmal die Relation „Stuhl auf Tisch". Somit ist die Relation „Stuhl vor Tisch" für einen Tisch sehr häufig und für einen Stuhl zumindest üblich. Die Relation „Stuhl auf Tisch" ist jedoch unüblich. Gegeben eine Anfrage, kann automatisch erkannt werden, welche Relationen für die Fallbasis unüblich sind und welche eventuell fehlen. Steht also in der Anfrage ein Stuhl auf einem Tisch und kein Stuhl davor, so kann der Benutzer vom System auf diese unübliche Situation hingewiesen werden. Als absolutes Entscheidungskriterium für korrekte oder fehlerhafte Situationen kann eine solche Statistik natürlich nicht dienen, da sie sonst alles Neue verhindern würde.

- Gibt es in Fall und Anfrage mehrere Objekte mit gleichen Eigenschaften, so wird beim „initial match" ein Strukturvergleich benötigt. Gibt es zum Beispiel in Anfrage und Fall mehrere Stühle und steht vor einem Stuhl der Anfrage ein Tisch, der vor dem Fenster steht, welches den südlichen Rand des Gebäudes berührt, so kann nur durch einen Strukturvergleich festgestellt werden, welcher Stuhl des Falles diesem Stuhl der Anfrage zugeordnet werden kann.

- Die Erkennung von Zusammenhängen liefert eine generische Heuristik für die Auswahl des zu übertragenden Teiles des Falles zwecks

Ergänzung einer Anfrage: *Nur Teile des Falles, die mit dem Ergebnis des „initial match" in Beziehung stehen, sollten übertragen werden.* Alles andere steht in keinem erkennbaren Zusammenhang mit dem in Anfrage und Fall gemeinsamen Teil. Diese Heuristik bietet die Möglichkeit, zum Zeitpunkt der Anfrage zu entscheiden, was in einem Fall Problemteil und was Lösungsteil ist. Entsprechend müssen die Fälle diese Information nicht direkt enthalten, was die Erzeugung der Fälle vereinfacht. Gegeben einen Bauplan, müßte man ohne diese Heuristik alle enthaltenen Probleme mit zugehörigen Lösungen in Fälle gruppieren. Dabei würden nicht nur Redundanzen entstehen, da die gleichen Objekte an verschiedenen Problemen und Lösungen beteiligt sein können, sondern es wäre auch eine sehr mühevolle und wahrscheinlich uninteressante Aufgabe. Außerdem könnte nicht sichergestellt werden, daß wirklich alle Probleme und Lösungen gruppiert wurden und die Fälle für beliebige Anfragen genügend Kontext enthalten. Natürlich besteht die Möglichkeit, anwendungsspezifisches Wissen zu modellieren, um ohne die Übertragungsheuristik Probleme und Lösungen zu unterscheiden und zu erkennen. Das dazu benötigte Wissen ist jedoch im allgemeinen schwer zugänglich und würde die Anwendbarkeit eines Systems auf die modellierten Bereiche einschränken.

Insbesondere die Frage, wieviel Kontext ein Fall benötigt, ist selten statisch zu entscheiden. Geht es zum Beispiel um die Planung des Zuluftsystems, so besteht eine sinnvolle Problembeschreibung eines Falles aus Zuluftstammleitungen und Zuluftauslaßöffnungen (wie zum Beispiel in Abbildung 2.3 des vorhergehenden Kapitels). Eine Lösung könnte aus den zur Verbindung der Auslaßöffnungen verwendeten Zuluftverbindungsleitungen bestehen. Zusätzlich werden die Objekte des Trägersystems des Deckenhohlraumes als Kontext benötigt. Enthält die Anfrage ebenfalls nur Objekte des Zuluft- und Trägersystems, so enthält der Fall genügend Kontext. Enthält jedoch die Anfrage zum Beispiel auch Objekte des Abluftsystems, so sollten diese auch im Kontext des Falles enthalten sein, um Kollisionen zu vermeiden. Je nach Anfrage müßten entsprechend verschiedene Objekte im Kontext eines Falles enthalten sein.

Beim strukturorientierten fallbasierten Schließen kann der gesamte Bauplan als ein Fall betrachtet werden. Gegeben eine Anfrage, wird der in Anfrage und Fall gemeinsame Teil durch Strukturvergleich bestimmt und dynamisch der Lösungsteil ausgewählt. So ist man sicher,

daß der entstandene Fall den benötigten Kontext enthält. In Abschnitt 6.6 wird ein Beispiel für die Benutzung eines Gebäudeplans als Fall beschrieben.

- Da der zu übertragende Lösungsteil aufgrund seiner Beziehungen zum in Anfrage und Fall gemeinsamen Teil ausgewählt wurde, sind die Beziehungen, die zwischen Anfrage und Lösungsteil gelten sollen, bekannt. Entsprechend können diese überprüft und, falls sie durch die Übertragung des Lösungsteiles aus dem Fall auf die Anfrage verletzt wurden, korrigiert werden. Würde man nicht strukturbasiert vorgehen und wollte aber trotzdem die Struktur des Falles nicht ungenutzt lassen, so müßte man spätestens an dieser Stelle Beziehungen im Fall erkennen und nach der Übertragung in der Anfrage rekonstruieren. Dies spricht jedoch dafür, gleich von vornherein strukturbasiert vorzugehen.

3.3 Das Konzept des strukturorientierten fallbasierten Schließens

Das in dieser Arbeit vorgestellte Konzept zum strukturorientierten fallbasierten Schließen läßt sich fast problemlos in den Zyklus von Aamodt (Abbildung 3.1) einordnen, wie Abbildung 3.2 zeigt. Der wichtigste Unterschied ist, daß das Konzept das von Aamodt erwähnte Wissen und die beschriebene Funktionalität genau spezifiziert (Wissen am Beispiel: Abschnitt 2.3, Wissen im allgemeinen: Abschnitt 7.2, Fallbasis: Abschnitt 7.1) und damit eine generische Anleitung zum Bau eines fallbasierten Systems liefert. Insbesondere für das zum „Reuse" benötigte Wissen, aber auch für die anderen strukturorientierten Teilschritte findet sich kein anderer, ähnlich leistungsfähiger.

Die Suche nach einem passenden Fall gliedert sich wie bei Aamodt in zwei Stufen. Im „Search"-Schritt wird durch Vergleichsverfahren geringer Komplexität als Vorauswahl eine kleine Menge von zur Anfrage ähnlichen Fällen bestimmt. Der aufwendigere „Initial Match" vergleicht die Anfrage erneut mit den Fällen der Vorauswahl um die passenden Fälle zu finden. Er liefert zusätzlich die gemeinsame Teilstruktur, deren Kenntnis Voraussetzung der Anpassung ist.

Bei der Beschreibung der Vergleichsfunktionen im nachfolgenden Kapitel wird deutlich, daß „ähnlich" ein Approximation von „passend" ist. Würde man also nur ähnliche Fälle suchen, zum Beispiel weil der Benutzer keine automatische Anpassung wünscht, wäre der aufwendige „Initial Match" zum

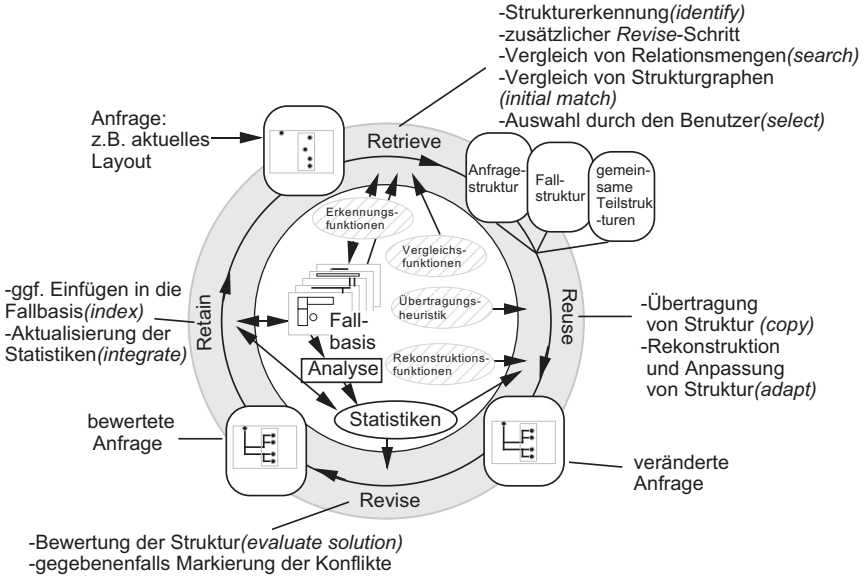

Abbildung 3.2: Einordnung des strukturorientierten fallbasierten Schließens in den Zyklus von [Aamodt und Plaza, 1994]. Zum Vergleich sind die von Aamodt benutzten Begriffe kursiv angemerkt.

Retrieval nicht nötig. Insofern könnte man den „Initial Match" als verbindenden Zwischenschritt zwischen Retrieval und Reuse bezeichnen.

Neben der genauen Spezifikation des benötigten Wissens wird gibt es einen weiteren wichtigen Unterschied zu Aamodt. Schon die Anfrage wird durch einen „Revise"-Schritt bewertet. Dies ermöglicht eine Korrektur der Anfrage, wozu bisher das fallbasierte Schließen nicht eingesetzt wurde. Abbildung 3.3 zeigt ein vereinfachtes Beispiel. Die Anfrage besteht aus einer Zuluftstammleitung (links oben) und mehreren Zuluftauslaßöffnungen. Durch die Statistik wird der Benutzer darauf hingewiesen, daß die zweite Auslaßöffnung von oben in unüblichen Beziehungen zu den anderen Auslaßöffnungen steht. Er markiert darauf die Position der Auslaßöffnung als falsch. Mit Hilfe eines passenden Falles können daraufhin in einem Schritt die Position der Auslaßöffnung korrigiert und die Verbindungsleitungen ergänzt werden. Die Abschnitte 3.4.2 und 3.5 werden die Methodik der Korrektur und Abschnitt 5.2 den Anpassungsalgorithmus explizit beschreiben.

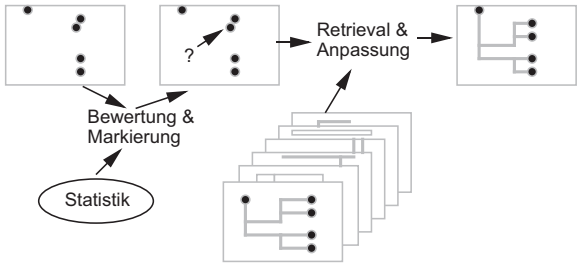

Abbildung 3.3: Ein Beispiel für gleichzeitige Korrektur und Ergänzung einer Anfrage.

Ein wichtiges Merkmal des im folgenden beschriebenen Problemlöseprozesses ist seine Interaktivität. Jedes Zwischenergebnis kann vom Benutzer betrachtet und zum Teil verändert werden. Dies ist sinnvoll, da alle Teilschritte auf Heuristiken basieren, die im Normalfall dem Benutzerwunsch entsprechen sollten, aber nicht zwingend sind. Deshalb ist es wahrscheinlich, daß der Benutzer in einigen Situationen den Problemlöseprozeß durch Veränderung der Zwischenergebnisse steuern möchte. Besitzt der Benutzer jedoch nicht die Erfahrung, um steuernd einzugreifen oder möchte er es aus anderen Gründen nicht, so kann das Verfahren auch ohne Benutzerinteraktion ablaufen.

3.3.1 Vorbereitende Schritte

Vor Beginn der Bearbeitung von Anfragen durch das SFBS wird das benutzte Wissen automatisch aufbereitet. Hierbei werden die in den Fällen geltenden Relationen erkannt und mit den beteiligten Objekten verknüpft. Die Erkennungsfunktionen für die einzelnen Relationstypen sind nicht Bestandteil des generischen Verfahrens, sondern werden durch das Modell, wie am Beispiel in Kapitel 2 gezeigt und in Abschnitt 7.2 formal beschrieben, bereitgestellt. Dadurch ist das Verfahren unabhängig vom Datenformat der Fälle und Anfragen.

Ein Analyseschritt berechnet die objektspezifischen Häufigkeiten von Relationen (vergleiche Abschnitt 3.2) und „relevanten" Objekteigenschaften und legt sie zur weiteren Verwendung in Statistiken ab. „Relevante" Objekteigenschaften sind solche, die durch die Rekonstruktionsfunktionen verändert werden können. Die Statistik wird bei der Rekonstruktion verwendet, um zu überprüfen, ob eine rekonstruierte Objekteigenschaft „üblich" ist (siehe Kapitel 5). Will der Benutzer bestimmte Relationen oder Objekteigenschaften verbieten oder erzwingen, so kann er die Statistik von Hand entsprechend verändern. Zum Beispiel kann er für Regale lieferbare Maße vorgeben oder die Häufigkeit der „Stuhl vor Tisch"-Beziehung mit 100% angeben. Letzteres hätte zu Folge, daß er auf jeden Tisch oder Stuhl, der in keiner solchen Beziehung steht, hingewiesen würde.

3.3.2 Retrieval mit integrierter Bewertung

Nach diesen vorbereitenden Schritten, die off-line durchgeführt werden und somit die Antwortzeit des SFBS nicht beeinflussen, können Anfragen bearbeitet werden. Zuerst wird, wie bei den Fällen der Fallbasis, die Struktur der Anfrage berechnet. Die erkannten Relationen werden dem Benutzer zusammen mit ihrer objektspezifischen Häufigkeit in der Fallbasis angezeigt. Als Reaktion kann er die Position von Objekten als „frei" markieren. Während des Retrievals und der Anpassung wird versucht, eine neue Position für ein freies Objekt zu finden und es dorthin zu verschieben (vergleiche Abbildung 3.3).

Beim Retrieval kann deshalb davon ausgegangen werden, daß die Anfragestruktur bis auf die freien Objekte korrekt ist und nicht korrigiert, sondern ergänzt werden soll. Dazu werden Fälle der Fallbasis gesucht, deren Strukturen möglichst große Teile der Anfragestruktur enthalten. Die Vergleiche der Relationsmengen und Strukturgraphen benutzen beide die vorgegebenen

Vergleichsfunktionen (vergleiche Abschnitt 2.3.2). Der Unterschied besteht in der Komplexität des Vergleichs und der Qualität des Ergebnisses. Der Vergleich zweier Relationsmengen kann in polynomieller Zeit, abhängig von der Größe der Relationsmengen, durchgeführt werden und liefert eine Obergrenze für die Größe einer in Anfrage und Fall gemeinsamen Struktur. Der Vergleich zweier Relationsgraphen besitzt hingegen eine exponentielle Komplexität, liefert aber nicht nur die exakte Größe der größten gemeinsamen Struktur, sondern die Struktur selbst. Die verschiedenen Möglichkeiten, die beiden Vergleiche hintereinander zu schalten beziehungsweise zu kombinieren, wird in Abschnitt 4.3 beschrieben. Die endgültige Entscheidung, welcher Fall zur Anpassung benutzt wird, bleibt jedoch dem Benutzer überlassen. Als Standard wählt das SFBS den Fall mit der größten mit der Anfrage gemeinsamen Teilstruktur. Die gemeinsame Teilstruktur enthält eine Zuordnung von zueinander kompatiblen Objekten und Relationen aus Fall und Anfrage.

3.3.3 Der Anpassungsschritt

Eingabeparameter der Anpassung sind die Anfragestruktur, die Struktur des ausgewählten Falles und deren gemeinsame Teilstrukturen. Die Anpassung ergänzt die Anfragestruktur um Teile der Struktur des Falles, die bisher nicht in der Anfrage enthalten waren, aber zu einer gemeinsamen Teilstruktur über Relationen verbunden sind. Gibt es mehrere gemeinsame Teilstrukturen, kann der Benutzer eine auswählen, oder es wird als Standard die größte gewählt. Gibt es mehrere gleich große Teilstrukturen, wird zufällig eine ausgewählt. Der Rekonstruktions- und Anpassungsschritt verändert die Anfrage, indem die Rekonstruktionsfunktionen der Relationen benutzt werden, um die Relationen der Fallstruktur in der Anfrage zu rekonstruieren. Die Rekonstruktion einer Relation entspricht dabei der Auswertung eines Constraints, des Gültigkeit in dem Fall erkannt wurde und daraufhin auch in der Anfrage gelten soll.

Ein Beispiel zeigt Abbildung 3.3. Zuerst wurden die Relationen und Verbindungsleitungen aus dem Fall in die Anfrage kopiert. Im Rekonstruktionsschritt wurden dann die neuen Positionen der Verbindungsleitungen und der markierten Auslaßöffnung unter Benutzung der Rekonstruktionsfunktionen und entsprechend der übertragenen Relationen berechnet.

3.3.4 Nachfolgende Schritte

Falls der Benutzer es wünscht, kann eine Bewertung des Ergebnisses durchgeführt und das Ergebnis in die Fallbasis eingefügt werden, woraufhin die Statistiken aktualisiert würden. Will man mehrere Fälle zur Anpassung benutzten, so kann man den Zyklus mehrere Male durchlaufen und jeweils die Teile auswählen, die man aus einem Fall übertragen lassen möchte. Die nachfolgenden Abschnitte werden die Methodik des Retrievals und der Anpassung ausführlicher beschreiben, insbesondere unter den verschiedenen Absichten der Ergänzung und Korrektur einer Anfrage.

3.4 Die Suche nach dem passenden Fall

Bei der Konzeption des Retrievals muß entschieden werden, zu welchem Zweck ein Fall gesucht wird. Zum einen kann ein Fall zur Korrektur oder als Alternative zu Teilen der Anfrage gesucht werden, und zum anderen kann man einen Fall zwecks einer Ergänzung oder Detaillierung der Anfrage suchen. Falls ein Fall sich zu beidem eignet, kann auf Basis dieses Falles auch eine Korrektur und eine Ergänzung in einem Schritt durchgeführt werden.

3.4.1 Passende Fälle zur Ergänzung

Wird nach einer Ergänzung der Anfrage gesucht, so wird derjenige Fall am einfachsten zur Anpassung benutzt werden können, der möglichst große Teile der Anfrage möglichst exakt enthält. Sind Teile der Anfrage nicht in einem Fall enthalten, so kann es bei der Anpassung zu Konflikten kommen, da diese Teile nicht zu den aus dem Fall übertragenen Teilen passen müssen. Die Größe des nicht im Fall enthaltenen Teiles der Anfrage kann also als Maß für das Risiko der Anpassung benutzt werden.

Ein anderer Konflikt kann entstehen, wenn der Fall Teile der Anfrage nur ungefähr enthält, der entsprechende Teil des Falles also nur ähnlich und nicht identisch zu einem Teil der Anfrage ist. Sicher wäre man, wenn man fordern würde, daß ein Fall die Anfrage vollständig und exakt enthält. Das macht jedoch nur Sinn, wenn für einen großen Teil der möglichen Anfragen solche Fälle in der Fallbasis enthalten sind. Das ist nur in sehr eingeschränkten Anwendungsgebieten möglich. Deswegen formuliert man Wissen, um mit Unterschieden zwischen Anfrage und Fall umzugehen. Dabei wird versucht, die Übertragung von Eigenschaften des Falles auf die Anfrage trotz der Un-

terschiede sicherzustellen, in dem man sie im Fall mittels des anwendungsspezifischem Wissens explizit erkennt und in der Anfrage rekonstruiert.

Zum Beispiel läßt die von TOPO benutzte Kompatibilitätsfunktion zu, daß sich kompatible Objekte in ihrer geometrischen Position und Größe unterscheiden. Um trotzdem die Objektpositionen aus dem Fall auf die Anfrage übertragen zu können, werden statt der exakten die relativen Objektpositionen und Objektgrößen explizit durch die Erkennungsfunktionen der räumlichen Relationen erkannt und in der Anfrage rekonstruiert. Trotzdem kann ein Qualitätsverlust eintreten. Das liegt daran, daß die benutzten räumlichen Relationen nicht exakt sind, also einige Unterschiede nicht differenzieren (siehe Abbildung 3.4). Theoretisch wäre auch eine Modellierung exakter Relationen möglich. Dann müßten jedoch die relativen Positionen der Objekte des Falles mit denen der Anfrage exakt übereinstimmen, um einander zugeordnet zu werden. Konkret hieße dies zum Beispiel, daß zwei „nahe bei" Relationen nur dann kompatibel sind, wenn die relative Distanz Δd exakt gleich ist, also zum Beispiel in beiden Relationen daß 0.7643-fache der maximalen Objektgröße der beteiligten Objekte. Damit wären nur sehr wenige Relationen kompatibel und damit könnten wiederum nur in stark eingeschränkten Anwendungsgebieten genügend Fälle bereitgestellt werden.

Andererseits wird derjenige Fall den größten Fortschritt ermöglichen, der möglichst viele und möglichst sinnvolle Ergänzungen des mit der Anfrage gemeinsamen Teiles bietet. Die vom Anwendungsgebiet unabhängige Übertragungsheuristik (vergleiche Abschnitt 3.3.3) erlaubt, die Menge der durch Anpassung eines Falles erzielbaren Ergänzungen abzuschätzen, in dem man die Zahl der zum gemeinsamen Teil in Beziehung stehenden Objekte betrachtet. Diese Zahl beschreibt also den durch einen Fall zu erwartenden Fortschritt.

Entsprechend den bisher vorgestellten Argumenten sollte also derjenige Fall zur Anpassung gewählt werden, der möglichst große Teile der Anfrage enthält und gleichzeitig viele Ergänzungen bietet. Genaugenommen sind diese beiden Anforderungen jedoch nur Heuristiken, die nur einen Hinweis auf die Eignung eines Falles liefern können. Um die Eignung eines Falles genau zu bestimmen, müßte man eine anwendungsspezifische Bewertung modellieren, die zum Beispiel entscheidet, wie gravierend die Unterschiede zwischen Fall und Anfrage und wie nützlich die Ergänzungen des Falles sind. Deswegen bietet TOPO zwar Vergleichsfunktionen an, die von einem Retrievalverfahren zum Retrieval benutzt werden, überläßt aber letztendlich die endgültige Auswahl dem Benutzer.

In TOPO wird die Auswahl des „besten" Falles deswegen letztendlich

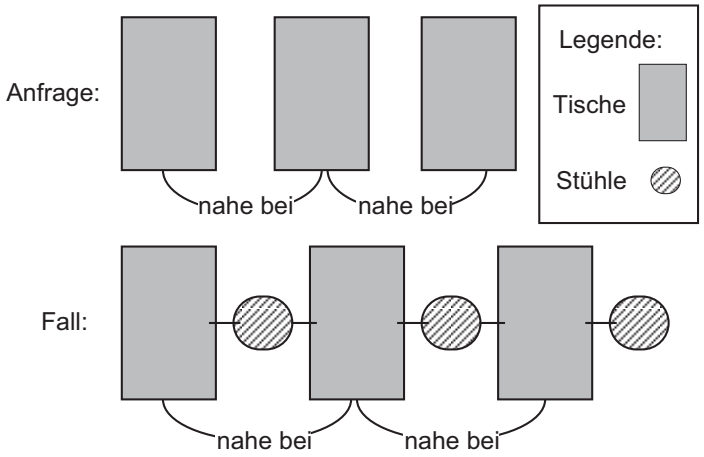

Abbildung 3.4: Die Tische stehen in Anfrage und Fall in identischen Beziehungen, da die Relation „nahe bei" den Abstand nicht exakt differenziert. Dies führt dazu, daß die Übertragung der Stühle zu Konflikten führt, die nur dadurch gelöst werden können, daß die Positionen der Tische in der Anfrage verändert werden.

dem Benutzer überlassen.

3.4.2 Passende Fälle zur Korrektur

Wird das Retrieval mit dem Ziel eingesetzt, eine Alternative zu Teilen der Anfrage zu finden, zum Beispiel, um sie zu korrigieren, so muß man entscheiden, welche Teile der Anfrage ersetzt werden sollen. Prinzipiell gibt es zwei Möglichkeiten zu erkennen, daß ein Teil der Anfrage korrigiert werden soll: Zum einen könnte ein Teil der Anfrage zu einem schlechten Fall ähnlich sein. Schlechte Fälle werden jedoch selten gespeichert, da keiner sie absichtlich erzeugt und sie auch keine konstruktive Hilfe leisten. TOPO benutzt die zweite Möglichkeit. Mithilfe der schon beschriebenen Statistik findet TOPO diejenigen Relationen der Anfrage, die in der Fallbasis, und somit in den guten Fällen selten sind, und weist den Benutzer auf diese hin. Der Benutzer kann daraufhin die Objekte freigeben, die an einer seltenen Relation beteiligt sind. In TOPO lassen sich nur die Position von Objekten und damit deren räumliche Beziehungen zur Ersetzung freigeben. Auf die gleiche Weise könnte man jedoch auch die semantischen Eigenschaften eines Objektes freigeben und

diese fallbasiert ersetzen.

Gibt man die Position eines Objektes frei, so ist es die Aufgabe des Retrievals, einen Fall zu finden, der möglichst große Teile der Anfrage enthält, aber das freigegebene Objekt in einer anderen Position als in der Anfrage. Um die Position eines freien Objektes nicht beliebig, sondern möglichst wenig zu verändern, benutzt TOPO folgende Regel: Sei A ein freies Anfrageobjekt, und es stehe zu den Objekten A_i in Beziehung. Seien F_i die den A_i zugeordneten Fallobjekte. Dann ordnet TOPO dasjenige Fallobjekt F dem Objekt A zu, das zu möglichst vielen Objekten F_i in Beziehung steht und überträgt die Relationen von F zu F_i auf A. Dadurch wird das freie Anfrageobjekt nach der Anpassung möglichst zu den gleichen Objekten wie vorher in Beziehung stehen, aber in den Beziehungen des Fallobjektes. Für die gemeinsame Teilstruktur bedeutet dies, daß an sich inkompatible Relationen einander zugeordnet werden. Aber es ist ja gerade die Absicht, daß die Relationen eines freigegebenen Objektes alternativen Relationen des Falles zugeordnet werden, um sie in der Anfrage durch diese zu ersetzen. In Abbildung 3.3 berühren sich zum Beispiel zwei Auslaßöffnungen. Die Relation wird als unüblich erkannt, und der Benutzer gibt die Position der unteren Auslaßöffnung zur Korrektur frei. Daraufhin wird die freie Auslaßöffnung der Anfrage derjenigen Auslaßöffnung des Falles zugeordnet, die im Fall auch in Beziehung zur obersten Auslaßöffnung steht, aber eben in einer anderen. Diese andere Beziehung wird aus dem Fall in die Anfrage übertragen und rekonstruiert.

3.5 Anpassung zwecks Korrektur und Ergänzung

Bis auf die freien Objekte werden alle Anfrageobjekte bei der Anpassung nicht verändert. Die Anpassung ergänzt die Anfragestruktur um diejenigen Relationen und Objekte, die zur gemeinsamen Teilstruktur, direkt oder über dritte, in Verbindung stehen. Gleichzeitig ersetzt die Anpassung die Relationen der freien Objekte durch die ihnen zugeordneten Relationen des Falles. Dieses relativ einfache Konzept wird durch das Matching ermöglicht. Dabei können zwei Probleme auftreten. Zum einen können unwichtige Zusammenhänge übertragen werden, die dazu führen, daß zu viele Objekte in der Anfrage erzeugt werden. Zum anderen kann die entstandene Struktur aufgrund der im vorherigen Abschnitt 3.4.1 erwähnten Unterschiede inkonsistent sein, wie in Abschnitt 3.5.2 beschrieben wird.

3.5.1 Maßnahmen gegen die Übertragung unwichtiger Zusammenhänge

Entsprechend der bisherigen Übertragungsheuristik würde ein Stuhl aus einem Fall aufgrund seiner räumlichen Beziehung zu einer Zuluftleitung im Fall auf eine Anfrage übertragen, falls die Zuluftleitung Bestandteil der gemeinsamen Teilstruktur ist. Dies wird im allgemeinen nicht gewollt sein, da zwischen Stuhl und Zuluftleitung kein direkter semantischer Zusammenhang besteht. Um dies zu berücksichtigen, kann man die Übertragungsheuristik um Wissen ergänzen, das die semantischen Zusammenhänge verschiedener Objekte beschreibt. Zu diesem Zweck ist ein Abhängigkeitsmodell für die im Anwendungsgebiet von TOPO vorkommenden Objekte unter dem Namen PM5 [Hovestadt, 1995] entstanden. Es besagt zum Beispiel, daß Zuluftauslaßöffnungen und Zuluftverbindungsleitungen semantisch zusammenhängen. Mit solchem Wissen kann auch bewirkt werden, daß überhaupt nur Relationen zwischen semantisch zusammenhängenden Objekten erkannt werden.

3.5.2 Entstehung und Lösung von Konflikten

Durch die Übertragung von Objekten und Relationen aus einem Fall auf eine Anfrage können zwei Arten von Konflikten enstehen. Zum einen können Konflikte zwischen den übertragenen und den in der Anfrage vorhandenen, aber nicht dem Fall zugeordneten Objekten entstehen. Dies führt häufig zu unüblichen Relationen, die im Schritt „Revise" mit Hilfe der durch Analyse erzeugten Statistiken erkannt werden. Da die am Konflikt beteiligten nicht zugeordneten Anfrageobjekte keinen Objekten des Falles zugeordnet werden konnten, kann der Fall nicht bei der Konfliktlösung helfen. Zur Korrektur des Konfliktes wird deshalb ein neuer Fall zur Korrektur gesucht, wie in Abschnitt 3.4.2 beschrieben.

Die zweite Art von Konflikten kann durch Unterschiede zwischen Fall und Anfrage entstehen, die durch die Relationen nicht repräsentiert werden. Hierfür zeigte schon Abbildung 3.4 ein Beispiel. Der dort beschriebene Konflikt läßt sich nur durch eine Freigabe der Tischpositionen lösen. Abbildung 3.5 zeigt ein weiteres Beispiel, das ohne Veränderung der Anfrage automatisch korrigiert werden kann. In diesem Beispiel sind die Wände des Büros in der Anfrage nur wenige Zentimeter weiter voneinander entfernt als im Fall. Das im Fall zusätzlich enthaltene Fenster „berührt" beide Wände. Nach der Übertragung des Fensters und der beiden Relationen auf die Anfrage ist die entstandene Struktur inkonsistent, da das Fenster zu schmal ist, um zwei

Wände des Büros der Anfrage zu berühren. Entsprechend muß relaxiert werden. Dafür gibt es verschiedene Möglichkeiten. Zum einen können die beiden Relationen relaxiert werden. Dies würde jedoch explizite Relaxationsregeln für die modellierten Relationen erfordern. Ohne relationsspezifische Relaxationsregeln kann man zufällig oder nach Rückfrage beim Benutzer eine Relation auswählen, die vollständig relaxiert wird und somit wegfällt. Eine dritte Möglichkeit wäre, das Fenster zu verbreitern, wozu man Wissen über zulässige Objekteigenschaften benötigt.

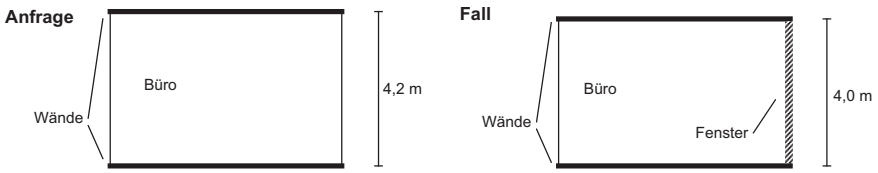

Abbildung 3.5: Die Wände des Büros sind in der Anfrage nur wenige Zentimeter weiter voneinander entfernt als im Fall. Trotzdem führt die Übertragung des Fensters zu Konflikten, die durch eine Verbreiterung des Fensters automatisch gelöst werden können.

TOPO berücksichtigt die letzten beiden Möglichkeiten. Für die zulässigen Objekteigenschaften wird jedoch kein explizit erhobenes anwendungsspezifisches Wissen benutzt, sondern die von der Analyse erstellte Statistik der in der Fallbasis vorkommenden Objekteigenschaften. Enthält die Statistik zum Beispiel ein entsprechend größeres Fenster, so wird das Fenster verbreitert. Enthält die Statistik kein passendes Fenster, so wird diejenige von beiden Relationen zur vollständigen Relaxation ausgewählt, die eine größere Distanz zur gemeinsamen Teilstruktur hat. Als Distanz wird die Anzahl der Objekte und Relationen betrachtet, die auf dem kürzesten Weg zu einem Objekt der gemeinsamen Teilstruktur liegen. Haben beide Relationen die gleiche Distanz, wie im Beispiel mit den Büros und dem Fenster, so wird zufällig eine Relation ausgewählt.

Ist anwendungsspezifisches Wissen zur Gewichtung und Relaxation von Relationen bekannt, so bietet es sich als Alternative zu dem bei TOPO gewählten Weg an, ausgefeilte Constraintlöser zur Rekonstruktion übertragener Strukturen einzusetzen. Als Beispiel wird das System IDIOM in Abschnitt 8.4.4 diskutiert.

Zusammenfassung der Antworten:

1. ⇒ Das fallbasierte Schließen sucht zur Anfrage ähnliche Fälle und benutzt diese, um die Anfrage zu verändern.

2. ⇒ Das SFBS besitzt die gleichen Teilschritte wie andere fallbasierte Verfahren, wobei der „initial match" als Zwischenschritt betrachtet werden kann, der nur von der Anpassung zwingend benötigt wird.

3. ⇒ Heuristiken, ergänzendes Wissen und Benutzerinteraktion:

   - ⇒ Unübliche Relationen werden durch Analyse der Fallbasis erkannt. Zusätzliches Wissen und/oder der Benutzer können übliche/unübliche Relationen vorgeben.
   - ⇒ Die Objekte der Anfrage werden als unveränderbar betrachtet. Der Benutzer kann jedoch Objekte in unüblichen Relationen freigeben.
   - ⇒ Der Fall mit größter gemeinsamer Teilstruktur wird gewählt. Der Benutzer kann jedoch einen beliebigen anderen Fall auswählen.
   - ⇒ Die größte gemeinsame Teilstruktur wird bei der Anpassung als Ausgangspunkt gewählt. Der Benutzer kann jedoch eine beliebige andere auswählen.
   - ⇒ Alles, was im Fall zur gewählten gemeinsamen Teilstruktur in Beziehung steht, wird übertragen. Diese Menge kann durch zusätzliches Wissen oder den Benutzer eingeschränkt werden.

4. ⇒ Bei der Ergänzung wird nach zur Anfrage identischen Strukturen gesucht. Bei Korrektur sollen freie Objekte der Anfrage im Fall in anderen Beziehungen stehen.

5. ⇒ Wenn die Relationen Unterschiede nicht exakt differenzieren oder die Anfrage nicht komplett im Fall enthalten ist, können Konflikte entstehen. Diese verursachen unübliche Relationen oder verhindern die Rekonstruktion von übertragenen Relationen in der Anfrage. Die Konflikte werden zum einen durch Relaxation von Relationen, und zum anderen durch einen erneuten SFBS-Zyklus gelöst.

Kapitel 4

Der Kern des Verfahrens: Der Vergleich von Strukturen

Fragen:

1. Wie kann man Strukturen als Graphen repräsentieren, und wie vergleicht man Graphen?
2. Wie kann man die Komplexität eines Graphvergleiches reduzieren?
3. Wie kann man Strukturen anders als durch Graphen repräsentieren und vergleichen, und welche Auswirkungen hat dies?
4. Wie vergleicht man transformierte Teilgraphen?
5. Was passiert bei kontinuierlichen Kompatibilitätsfunktionen für Objekte und Relationen?
6. Wie kann man mit abschätzenden Vergleichsfunktionen beim Retrieval effizient umgehen?

Wie im vorherigen Kapitel deutlich wurde, beeinflußt der Vergleich nicht nur die Suche nach geeigneten Fällen. Da der zweite Schritt der Suche („initial match"), ebenfalls die Zuordnung der Objekte und Relationen zwischen Fall und Anfrage bestimmt und diese Zuordnungen die Basis für die Übertragungsheuristik bilden, sollte die Vergleichsfunktion für Strukturgraphen sehr sorgfältig entworfen werden. Folgenden Kriterien sollte die Vergleichsfunktion genügen:

Um auch in großen Fallbasen suchen zu können, sollte der Vergleich eine geringe Komplexität besitzen.

Es sollen diejenigen Fälle gefunden werden, deren mit der Anfrage gemeinsame Teilstruktur möglichst groß ist (siehe Abschnitt 3.4.1).

Die Vergleichsfunktion sollte die größte gemeinsame Teilstruktur finden, um die Anpassung zu ermöglichen.

Leider gibt es einen Widerspruch zwischen der ersten und den letzten beiden Anforderungen, da der exakte Vergleich von Strukturen im allgemeinen np-vollständig ist. Dies Problem gehe ich von zwei Seiten an. Zum einen werde ich in Abschnitt 4.1 ein bekanntes Graphmatchingverfahren vorstellen und mehrere Möglichkeiten beschreiben, seine Effizienz zu steigern. Weitere Verbesserungen werden in Abschnitt 4.2.2 diskutiert. So wird zum Beispiel eine Idee einer anwendungsspezifischen Erweiterung der Relationsmenge vorgestellt, die die Effizienz des Vergleichs steigert.

Zum anderen beschreibt Abschnitt 4.3 einen Ansatz, das Ergebnis des „initial match" im „search"-Schritt durch schnellere Vergleichsfunktionen zu approximieren.

4.1 Graphmatching

Dieser Abschnitt befaßt sich mit der Repräsentation von Strukturen durch Graphen und der Suche nach dem größten gemeinsamen Teilgraphen zweier Graphen. Er basiert auf der in Kapitel 2 beschriebenen Modellierung von kompatiblen Objekten und Beziehungen. Das Ergebnis eines solchen Vergleichs, der größte gemeinsame Teilgraph, wird Matching genannt und besteht aus einer Menge von paarweisen Zuordnungen zwischen Knoten beider Graphen, die in beiden Graphen in kompatiblen Beziehungen stehen.

Die Abschnitte 4.1.1 bis 4.1.4 beschreiben das von mir gewählte Graphmatchingverfahren. Weitere Konzepte, die die Qualität des Ergebnisses verbessern und die Effizienz des Strukturvergleiches steigern, werden in Abschnitt 4.2 vorgestellt.

4.1.1 Kanten- statt Knotengraphen

Strukturen lassen sich auf zwei unterschiedliche Arten in Graphen abbilden. Je nachdem, welche Möglichkeit gewählt wird, liefert der Vergleich zweier Strukturen unterschiedliche Ergebnisse. Die eine Möglichkeit ist, die Objekte

einer Struktur als Knotenmenge des Graphen zu benutzen und die zwischen den Objekten geltenden Beziehungen als Kantenmenge. Den so entstandenen Graphen nennt man Knotengraph. Bei der anderen Möglichkeit bilden die Beziehungen die Knoten und die Objekte die Kanten. Einen so entstandenen Graphen nennt man Kantengraph.

Um meine Entscheidung für eine von beiden Möglichkeiten zu begründen, zeige ich die Auswirkungen auf das Ergebnis des Vergleichs zuerst am Beispiel und fasse sie dann theoretisch zusammen.

Bei dem Vergleich zweier Strukturen wird ein Matching der Knoten zweier Graphen berechnet. Die zugeordneten Knoten müssen in beiden Graphen durch kompatible Kanten verbunden sein.

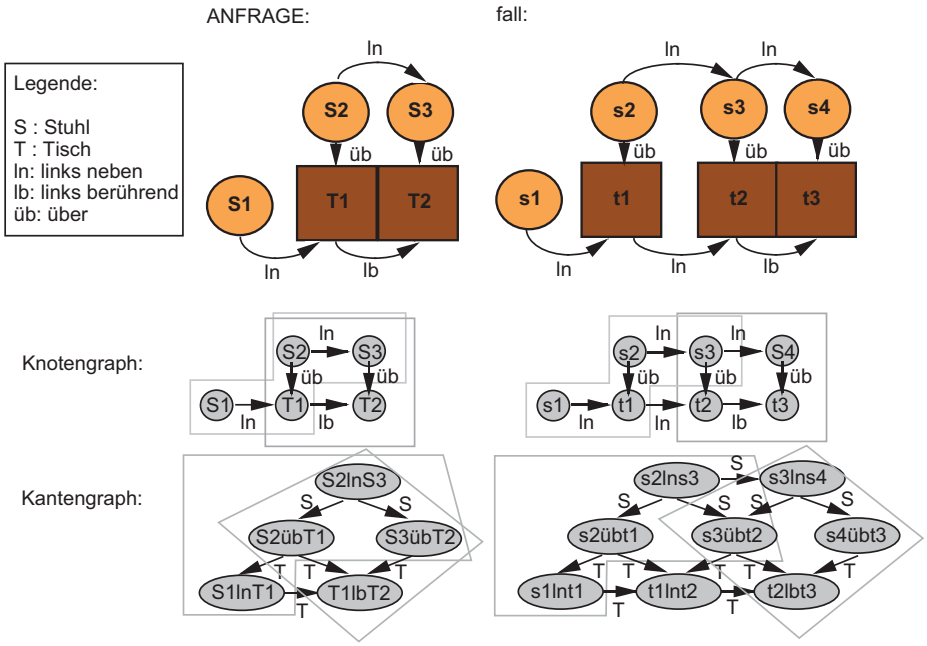

Abbildung 4.1: Die obere Zeile zeigt zwei räumliche Layouts, die aus durchnumerierten Stühlen und Tischen bestehen. Die Objekte der Anfrage sind durch Großbuchstaben, die des Falles durch Kleinbuchstaben bezeichnet. Die unteren Zeilen zeigen zunächst die Knoten- und dann die Kantengraphen, durch die beide Layouts beschrieben werden können. Die grauen Rahmen kennzeichen die größten in beiden Graphen einer Zeile enthaltenen gemeinsamen Teilgraphen.

In Abbildung 4.1 findet sich die gesamte Anfrage im Fall wieder bis darauf, daß sich im Fall die beiden Tische t1 und t2 nicht berühren. Dies verhindert im Matching der Knotengraphen, daß die Tische T1 und T2 den Tischen t1 und t2 zugeordnet werden können, da alle zugeordneten Knoten in beiden Graphen durch kompatible Kanten verbunden sein müssen. Die beiden größten Matchings enthalten die Zuordnungen {(S1,s1), (S2,s2), (S3,s3), (T1,t1)} und {(S2,s3), (S3,s4), (T1,t2), (T2,t3)}(vgl. graue Rahmen in Abbildung 4.1).

Matcht man hingegen die Kantengraphen, so verhindert die unterschiedliche Beziehung der Tische (T1,T2) und (t1,t2) nicht deren Zuordnung. Die Knoten der Kantengraphen repräsentieren Relationen. Die Relationsknoten sind durch gemeinsame Objekte verbunden. In dem durch den Vergleich der Kantengraphen erhaltenen Matching ist die Zuordnung beider Tische in den Zuordnungen der Relationen (S1lnT1, s1lnt1) und (S3übT2, s3übt2) enthalten.

Dieser Unterschied im Ergebnis des Vergleichs kann folgendermaßen zusammengefaßt werden. Ein Anfrageobjekt ist in dem Matching zweier Kantengraphen enthalten, wenn es mindestens *eine* Beziehung zu einem anderen im Matching enthaltenen Objekt besitzt, die zu der Beziehung im Fall kompatibel ist. Um im Matching zweier Knotengraphen enthalten zu sein, müssen *alle* Beziehungen zu anderen im Matching enthalten Objekte zu den Beziehungen des Falles kompatibel sein.

Ich habe mich für die Repräsentation von Strukturen durch Kantengraphen entschieden, weil ich den Vergleich von Knotengraphen für zu restriktiv halte. Der Vorteil, daß auf diese Weise häufig mehr Objekte zugeordnet werden können als beim Matching von Knotengraphen, wiegt aus meiner Sicht schwerer, als die Konflikte, die daraus entstehen können, daß beim Matching der Kantengraphen sich einige Relationen zugeordneter Objekte unterscheiden dürfen. Das im weiteren beschriebene Verfahren arbeitet jedoch unabhängig davon, ob man sich für Knoten- oder Kantengraphen entscheidet.

4.1.2 Erzeugung des Kompatibilitätsgraphen

Um die Mengen von Zuordnungen zwischen Knoten zweier Graphen zu finden, so daß einander zugeordnete Knoten in beiden Graphen in kompatiblen Beziehungen stehen, schlagen [Barrow und Burstall, 1976] die Erzeugung eines Kompatibilitätsgraphen vor. Dieser Kompatibilitätsgraph enthält alle möglichen „kompatiblen" Zuordnungen als Kompatibilitätsknoten (vgl.

Abbildung 4.2). Eine Zuordnung zweier Knoten ist „kompatibel", wenn die durch den Knoten repräsentierten Beziehungen im Sinne des vorherigen Abschnitts kompatibel sind. Der Kompatibilitätsknoten S2lnS3-s2lns3 repräsentiert zum Beispiel die kompatible Zuordnung der beiden Beziehungen S2lnS3 und s2lns3 zueinander.

Zwei Kompatibilitätsknoten können in drei verschiedenen Beziehungen zueinander stehen:

1. Sie können sich widersprechen, da die Knoten der durch beide Kompatibilitätsknoten repräsentierten Zuordnungen in beiden Ursprungsgraphen in unterschiedlichen Beziehungen stehen. Ein Beispiel sind die Kompatibilitätsknoten S2lnS3-s2lns3 und S3übT2-s4übt3. In der Anfrage stehen die Knoten S2lnS3 und S3übT2 über eine Stuhlkante in Beziehung, während s2lns3 und s4übt3 im Fall in keiner direkten Beziehung zueinander stehen. Deswegen existiert im Kompatibilitätsgraphen keine Kante zwischen den beiden Kompatibilitätsknoten.

2. Die zugeordneten Knoten können in beiden Ursprungsgraphen in der gleichen Beziehung stehen. Zum Beispiel stehen die Knoten S2lnS3 und S2übT1 der Anfrage in der gleichen Beziehung zueinander, wie die Knoten s2lns3 und s3übt2. Deswegen sind die Zuordnungen S2lnS3-s2lns3 und S2übT1-s3übt2 im Kompatibilitätsgraphen über eine dunkelgraue Kante verbunden.

3. Die hellgrauen Kanten verbinden Zuordnungen, die sich auf eine andere Weise nicht widersprechen. Sie widersprechen sich nicht, weil die zugeordneten Knoten in beiden Ursprungsgraphen in keiner Verbindung zueinander stehen. Dies gilt zum Beispiel für die Zuordnungen S2lnS3-s2lns3 und T1lbT2-t2lbt3. Der Knoten S2lnS3 ist nicht mit T1lbT2 verbunden und s2lns3 ebenfalls nicht mit t2lbt3.

Im folgenden nenne ich die dunkelgrauen Kanten „1-Kanten", da sie wegen einer vorhandenen Beziehung erzeugt wurden, und die hellgrauen Kanten „0-Kanten", da sie aufgrund einer nicht vorhandenen Beziehung erzeugt wurden. Ihre Unterscheidung wird in der von mir vorgeschlagenen Effizienzsteigerung des Graphmatchings eine wesentliche Rolle spielen. Dabei ist es irrelevant, ob man die in den Kompatibilitätsgraphen abgebildeten Strukturgraphen als Kanten- oder als Knotengraphen repräsentiert.

In dem Kompatibilitätsgraphen sind also je zwei Kompatibilitätsknoten durch eine Kante verbunden, wenn die durch sie repräsentierten Zuordnun-

gen sich nicht widersprechen. Dem größten Matching der beiden Ursprungsgraphen entspricht nun die größte miteinander vollständig verknüpfte Menge von Kompatibilitätsknoten, auch „größter vollständiger Teilgraph" genannt. Der größte vollständige Teilgraph des Kompatibilitätsgraphen entspricht also der größten Menge von sich nicht widersprechenden Zuordnungen.

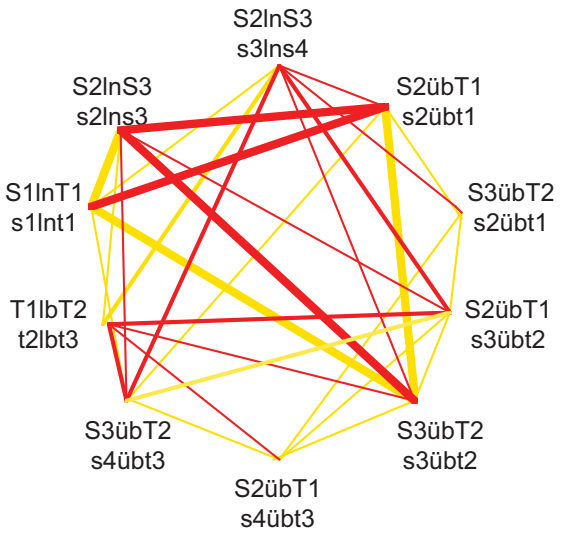

Abbildung 4.2: Diese Abbildung zeigt einen Kompatibilitätsgraphen. Seine Kompatibilitätsknoten repräsentieren alle „kompatiblen" Zuordnungen der Knoten der Kantengraphen aus Abbildung 4.1. Der Kompatibilitätsknoten S1lnT1-s1lnt1 repräsentiert zum Beispiel die Zuordnung der beiden im Namen genannten Knoten. Zwei Kompatibilitätsknoten (Zuordnungen) sind verbunden genau dann, wenn die zugeordneten Relationen in beiden Graphen keine Beziehung (hellgraue Kanten) oder die kompatible Beziehung (dunkelgraue Kanten) haben.

Abbildung 4.3 zeigt die beiden größten vollständigen Teilgraphen. Beide sind gleich groß, da sie jeweils vier Zuordnungen von Beziehungen enthalten. Unter dem Aspekt der Anzahl der zugeordneten Objekte ist die obere Zuordnung jedoch größer, da sie fünf Objekte enthält.

Vollständig verknüpfte Kompatibilitätsknotenmengen, die man nicht mehr erweitern kann, ohne daß unverknüpfte Kompatibilitätsknoten hinzukommen, nennt man „Cliquen". Der nächste Abschnitt wird sich mit der Suche nach solchen Cliquen in Graphen beschäftigen.

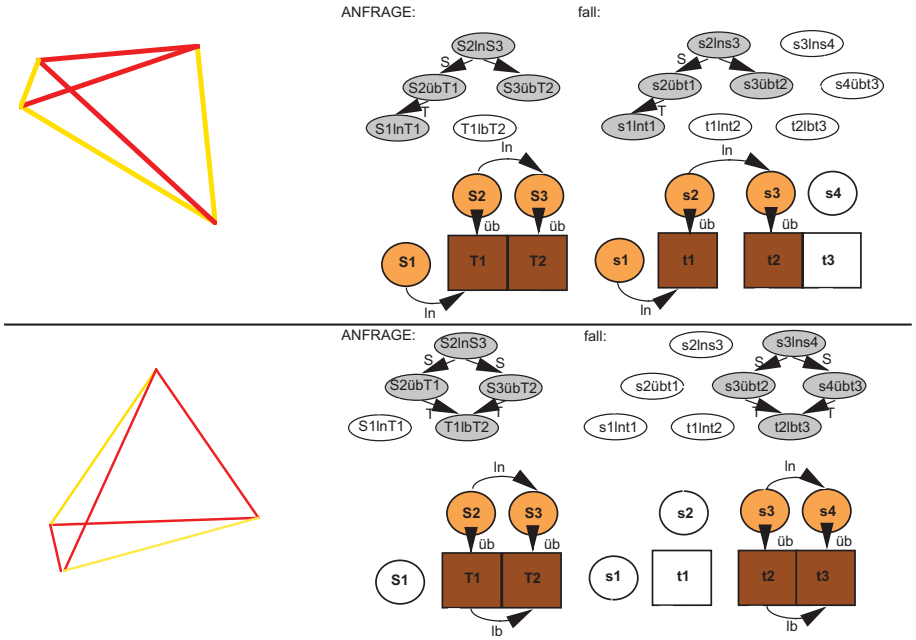

Abbildung 4.3: Die beiden linken Graphen zeigen die beiden größten vollständigen Teilgraphen des Kompatibilitätsgraphen. Rechts daneben sind die zugeordneten Knoten in den Ursprungsgraphen markiert. Unter jedem Graphen sieht man die entsprechenden zugeordneten Teile der Layouts.

4.1.3 Cliquensuche

Zur Suche nach Cliquen habe ich den Algorithmus von [Bron und Kerbosch, 1973] (im weiteren „max-clique$_{BK}$" genannt) gewählt. Er ist vollständig, findet folglich alle Cliquen eines Graphen, und ich habe keinen anderen vollständigen Algorithmus gefunden, welcher effizienter sein soll. Er beginnt mit allen einelementigen Teilgraphen und erweitert sie sukzessive durch mit dem bisherigen Teilgraphen komplett verbundene Kompatibilitätsknoten. Läßt sich ein Teilgraph nicht mehr erweitern, so ist eine Clique gefunden. Auf diese Weise zählt max-clique$_{BK}$ alle vollständigen Teilgraphen und damit auch alle Cliquen auf.

Abbildung 4.4 zeigt den Pseudo-Code des rekursiven Algorithmus. Der zu durchsuchende Graph sei gegeben durch $G := (V, E)$, wobei V die Knotenmenge[1] und E die Kantenmenge des Graphen sei. Das Verfahren max-clique$_{BK}$ unterscheidet nicht zwischen 1-Kanten und 0-Kanten. Die Parameter C, P und S bestehen aus Knotenmengen von V. Der Algorithmus wird mit $C = \emptyset, P = V$, und $S = \emptyset$ aufgerufen. Während des Ablaufs enthält C die Menge der Knoten des aktuell untersuchten vollständigen Teilgraphens. Alle potentiellen Erweiterungen von C, also alle Knoten aus V, die mit allen Knoten in C verbunden sind, sind entweder in P oder in S enthalten. Der Algorithmus fügt während der Suche alle Knoten aus P nacheinander zu C hinzu. War ein Knoten schon einmal in C enthalten, so wird er der Menge S hinzugefügt. Die Menge S enthält also schon benutzte Erweiterungen von C.

Die Rekursion wird in Zeile 1 unterbrochen, wenn $P = \emptyset$ und $S = \emptyset$. In diesem Fall ist C eine neue Clique, da es keine möglichen Erweiterungen der in C enthaltenen vollständigen verknüpften Knotenmenge mehr gibt. Die Clique ist neu, da nur Knoten aus P zur Erweiterung benutzt wurden, die Knoten alter Cliquen aber in S liegen. Gilt $P = \emptyset$ und $S \neq \emptyset$, so bricht die Rekursion auch ab, da die FOR-Schleife (Zeile 4) nicht durchlaufen wird. In diesem Fall enthält C zwar die Knoten eines vollständigen Teilgraphen, er könnte aber aus S erweitert werden. Somit ist C keine Clique, sondern Teilgraph einer alten Clique.

Die Bedingung für die Hinzunahme eines Knoten aus P zu C in 4.1 dient ausschließlich der Effizienzsteigerung. Bron und Kerbosch haben sich überlegt, daß alle Knoten, die zu einem gewählten u_i verbunden sind, im weiteren Verlauf der Suche sowieso zu C hinzugenommen werden und somit

[1] Bei der weiteren Beschreiben der Cliquensuche sind mit Knoten immer Kompatibilitätsknoten gemeint.

MAX-CLIQUE$_{BK}$(C,P,S)

1. IF P=$\emptyset$ AND S=$\emptyset$ THEN RETURN C
2. Sei P die Menge $\{u1, ..., u_k\}$
3. Sei u_t ein ausgewählter Knoten aus P
4. FOR i = 1 TO k DO

   4.1 IF $\{u_i, u_t\} \notin E$ THEN

   4.1.1 P := P$\backslash u_i$

   4.1.2 N := $\{v \in V \mid \{u_i, v\} \in E\}$ *(Nachbarn von u_i)*

   4.1.3 MAX-CLIQUE$_{BK}$ (C $+u_i$, P $\cap$ N, S $\cap$ N)

   4.1.4 S := S$+u_i$

Abbildung 4.4: Der Algorithmus max-clique$_{BK}$

nicht ohne u_i betrachtet werden müssen. Für die geschickte Auswahl von u_i gibt es verschiedene Strategien, die in [Koch *et al.*, 1996] beschrieben und diskutiert werden.

4.1.4 Effizienzsteigerung

Da der Algorithmus max-clique$_{BK}$ nicht zwischen 1-Kanten und 0-Kanten unterscheidet, erweitert er C auch um Knoten, die nur durch 0-Kanten mit C verbunden sind. Für die Menge der durch die Knoten repräsentierten Zuordnungen heißt dies, daß die neu hinzugenommen Knoten in *keiner* Beziehung zu den bisher zugeordneten Knoten stehen können.

Betrachte man zum Beispiel zwei Konzertsäle K1 und K2 mit je 1000 Stühlen, von denen die meisten Stühle mit acht anderen durch räumliche Nachbarschaft in Beziehung stehen (siehe etwa links oben in K1 in Abbildung 4.5). Beinhalteten die bisherigen Zuordnungen zum Beispiel Stühle aus den vorderen Reihen, so könnte max-clique$_{BK}$ eine Zuordnung des hinteren linken Stuhles aus K1 zu dem hinteren rechten Stuhl aus K2 zur bisherigen Zuordnungsmenge hinzunehmen, da sie über 0-Kanten zu allen Knoten der bisherigen Zuordnung verbunden sind. Dies liegt daran, daß *beide* Stühle *nicht* zu den bisher zugeordneten Stühlen der vorderen Reihen in Verbindung stehen. Diese neue Zuordnung wird jedoch selten zum größten

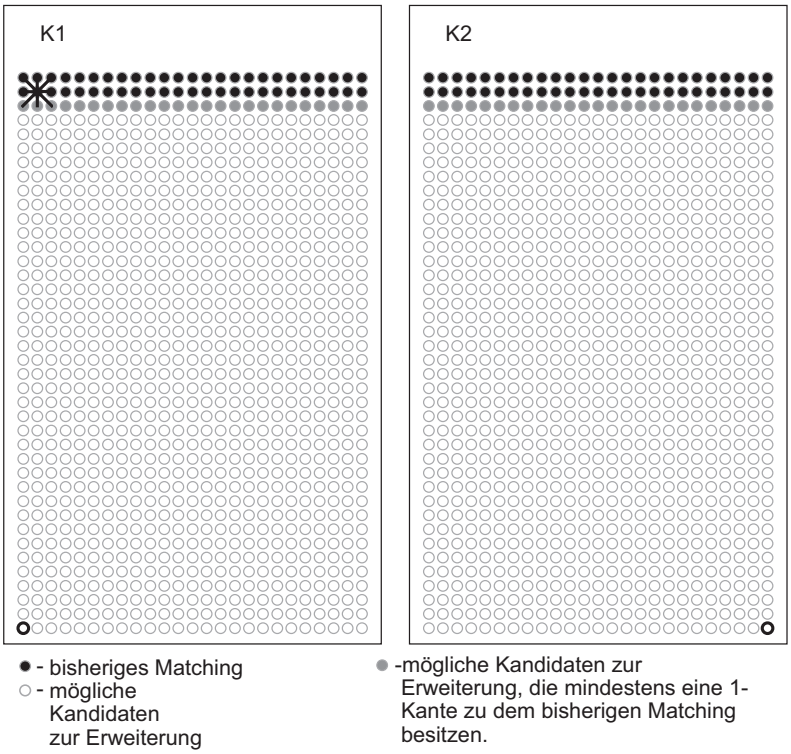

Abbildung 4.5: Das Beispiel des Vergleichs der Bestuhlung zweier Konzertsäle verdeutlicht die Auswirkung der Beachtung von 1-Kanten durch den Algorithmus. Erweitert man Matchings nur um Objekte, die über 1-Kanten zum bisherigen Matching in Verbindung stehen, so kann man viele überflüssige Vergleiche vermeiden.

MAX-CLIQUE$_{BK+}$(C, P1, P0, S)

1. IF P1=$\emptyset$ AND S=$\emptyset$ THEN RETURN C
2. Sei P1 die Menge $\{u1, ..., u_k\}$
3. Sei u_t ein ausgewählter Knoten aus P1
4. FOR i = 1 TO k DO

   4.1 IF $\{u_i, u_t\} \notin E1$ THEN

   4.1.1 P1 := P1$\backslash u_i$

   4.1.2 N1 := $\{v \in V \mid \{u_i, v\} \in E1\}$

   4.1.3 N0 := $\{v \in V \mid \{u_i, v\} \in E0\}$

   4.1.4 IF C = $\emptyset$

   THEN P1' := N1

   P0' := N0

   ELSE P1' := ((P1 $\cap$ ($N1 \cup N0$)) $\cup$ (P0 $\cap$ N1)

   P0' := P0 $\cap$ N0

   4.1.5 MAX-CLIQUE$_{BK}$ (C $+u_i$, P1', P0', S $\cap$ N1)

   4.1.6 S := S$+u_i$

Abbildung 4.6: Der Algorithmus max-clique$_{BK+}$

vollständigen Teilgraphen gehören, da der hintere linke Stuhl keine linken Nachbarn hat und der hintere rechte keine rechten. Für die Verwendung des Algorithmus zur Suche nach der größten Menge von kompatiblen Zuordnungen ist es daher sinnvoll, C nur um solche Zuordnungen zu erweitern, die mit mindestens einer 1-Kante zu C in Verbindung stehen. Das hat zur Folge, daß nur diejenigen Zuordnungen zu C hinzugenommen werden, deren Knoten zu mindestens einem der bisher zugeordneten Knoten in Beziehung stehen. Im Beispiel würden also nur solche Zuordnungen hinzugenommen, deren Stühle zu mindestens einem der bisher zugeordneten Stühle in K1 und K2 in Beziehung stehen. Abbildung 4.6 zeigt den modifizierten Algorithmus max-clique$_{BK+}$.

Da nun 0-Kanten und 1-Kanten unterschieden werden sollen, sind die Kanten des zu durchsuchenden Graphen in E1 für die 1-Kanten und E0 für die 0-Kanten unterteilt. Entsprechend ist die Parameterliste erweitert wor-

den. P1 enthält nun diejenigen potentiellen Erweiterungen, die über mindestens eine 1-Kante mit C verbunden sind, und P0 die anderen. Aufgerufen wird der Algorithmus mit P1=V und $P0 = \emptyset$. Da nur Kanten aus P1 hinzugenommen werden sollen, bricht der Algorithmus in Zeile 1 ab, wenn $P1 = \emptyset$. Genau wie P wurde auch die lokale Variable N in N1 und N0 unterteilt. Die Nachbarn des zur Erweiterung gewählten Knotens u_i werden also unterteilt, je nachdem, ob sie mit einer 1-Kante oder einer 0-Kante mit u_i verbunden sind. Die Abfrage in Zeile 4.1.4 dient dazu, die Mengen P0' und P1' vor der ersten Rekursion zu initialisieren. Vor der ersten Rekursion gilt $C = \emptyset$, und P0' und P1' werden die Mengen N0 und N1 zugewiesen. Komplizierter ist die Berechnung bei den tieferen Rekursionen. Die neue Menge P1' besteht aus den Knoten von P1, die mit u_i, und damit immer noch mit allen Knoten aus C verbunden sind. Hinzu kommen die Knoten aus P0, die über eine 1-Kante mit u_i verbunden sind und somit mit einer 1-Kante zum um u_i erweiterten C. P0' besteht aus denjenigen Knoten aus P0, die zum um u_i erweiterten C immer noch vollständig, aber nur über 0-Kanten verbunden sind, also P0 $\cap$ N0.

4.1.5 Branch-and-Bound Erweiterung

Um zu verhindern, daß Suchzweige verfolgt werden, die nicht zu einer größten Clique führen können, läßt sich der Algorithmus max-clique$_{BK+}$ um ein Branch-and-Bound-Verhalten erweitern. Dieses soll eine Suche in einem Zweig abbrechen, wenn die aktuell untersuchte Menge C zusammen mit all ihren möglichen Erweiterungen, also den Mengen P0 und P1, kleiner ist als die größte bisher gefundene Clique.

Um dieses Verhalten zu erzielen, braucht man sich nur die Größe MAX-C der bisher größten Clique zu merken und in Schritt 4.1.5 max-clique$_{BK+}$ nur dann aufrufen, wenn die neuen Mengen C, P0 und P1 zusammen größer als MAX-C sind. Wird diese Erweiterung benutzt, findet max-clique$_{BK+}$ nicht mehr unbedingt alle Cliquen. Somit können dem Benutzer nicht mehr alle durch die Cliquen repräsentierten Matchings angeboten werden, sondern nur das größte. Man kann diesen Effekt allerdings abschwächen, indem man nicht auf „> MAX-C", sondern auf „$\geq$ MAX-C" oder „$\geq$ (MAX-C - ϵ)" testet.

4.1.6 Theoretische Bewertung der Effizienzsteigerung

Seien S_1 und S_2 zwei zu vergleichende Strukturen. Der Kombinationsgraph enthält als Knoten alle möglichen kompatiblen Zuordnungen der Relationen

beider Strukturen. Offensichtlich ist die Größe des maximalen Matchings und somit die Größe einer maximalen Clique im Kombinationsgraphen durch die Anzahl der Relationen der kleineren Struktur beschränkt:

$$clique_{max} \leq min(AnzahlRelationen(S_1), AnzahlRelationen(S_2))$$

Die Verfahren max-clique$_{BK}$ und max-clique$_{BK+}$ zählen alle vollständigen Teilgraphen des Kombinationsgraphen auf. Entsprechend den obigen Erläuterungen gibt es maximal $clique_{max}$ Erweiterungsschritte, und die Anzahl der möglichen Erweiterungen hängt in jedem Schritt von der durchschnittlichen Anzahl der Nachbarn der im bisherigen Teilgraphen enthaltenen Knoten ab. Bezeichnet man diese durchschnittliche Anzahl mit $N(k)$, so hängt die Laufzeit der beiden Algorithmen von $N(k)^{clique_{max}}$ ab.

Bei max-clique$_{BK+}$ werden die Erweiterungen auf Nachbarn beschränkt, die mit mindestens einem der im bisherigen Teilgraphen enthalten Knoten durch eine 1-Kante verbunden sind. Wie man am Beispiel des Vergleiches der beiden Konzertsäle (Abbildung 4.5) sieht, kann der Anteil der 1-Kanten sehr gering sein, wodurch die Anzahl der möglichen Erweiterung $N(k)$ ebenfalls stark reduziert wird. In Anwendungen mit räumlichen Nachbarschaftsbeziehungen als Relationen wird der Anteil der 1-Kanten im allgemeinen desto kleiner sein, je größer die zu vergleichenden räumlichen Situationen sind. Dies liegt daran, daß in solchen Situationen die meisten Relationen über Objekte nicht miteinander verbunden sind und es somit sehr viel mehr 0-Kanten als 1-Kanten geben wird (vergleiche Abschnitt 4.1.2).

Die bisherigen Erläuterungen geben keine exakte Abschätzung, sondern eher einen Hinweis auf die Wirksamkeit der Veränderung von max-clique$_{BK}$ zu max-clique$_{BK+}$. Es scheint, daß die Veränderung des Algorithmus besonders beim Vergleich größerer Fälle wirksam ist. Dies zeigen auch die Beispiele in Abschnitt 6.6.

Um die Steigerung der Effizienz genauer zu untersuchen, müßte man den Anteil der vollständigen Teilgraphen abschätzen, die über 1-Kanten verbunden sind. Dadurch wäre bekannt, wieviele der von max-clique$_{BK}$ betrachteten Teilgraphen auch von max-clique$_{BK+}$ untersucht werden. In [Bender *et al.*, 1990] findet sich die Aussage, daß es bisher keine exakte Abschätzung der Anzahl der zusammenhängenden Graphen mit n Knoten und q Kanten gibt. Bei von TOPO durchgeführten Graphvergleichen war die Anzahl der 1-Kanten des Kompatibilitätsgraphen meistens wesentlich geringer als die Anzahl der 0-Kanten, wodurch max-clique$_{BK+}$ wesentlich schneller war als max-clique$_{BK}$. Einige Beispiele finden sich in Abschnitt 6.6 in Tabelle 6.21.

4.2 Weitere Verbesserungen und Aspekte des Strukturvergleiches

In diesem Abschnitt werden einige Aspekte des Strukturvergleiches diskutiert, die von der Technik des Graphvergleichs weitgehend unabhängig sind. Zum einen die Frage, wie man „verdrehte" oder „gespiegelte" Teile von Bauplänen erkennt und wie man im allgemeinen mit derartigen Transformationen umgeht. Der zweite Teil beschäftigt sich mit der Repräsentation markanter komplexer Relationen, wie zum Beispiel einer Reihe von Räumen. Durch die Verwendung solcher komplexer Relationen kann die Komplexität des Vergleichs zweier Graphen unter Umständen stark gesteigert werden werden. Der letzte Teilabschnitt beschäftigt sich mit der Verarbeitung kontinuierlicher Kompatibilitätsmaße, die nicht nur testen, ob zwei Objekte kompatibel sind, sondern auch Ähnlichkeiten zwischen an sich inkompatiblen Objekten und Relationen berücksichtigen.

4.2.1 Transformierte Teilmengen

Durch die Drehung oder Spiegelung von Teilen von Bauplänen ändert sich die Repräsentation der räumlichen Struktur, aber nicht unbedingt ihre Bedeutung. Steht in einem Fall zum Beispiel eine Reihe von Stühlen in x-Dimension vor einer Reihe von Tischen, so steht sie nach einer Drehung um 90° in y-Dimension vor der Tischreihe. Wird dieser Fall zur Ergänzung einer Anfrage, bestehend aus einer Reihe von Tischen, benutzt, so sollte die Drehung die Nützlichkeit des Falles nicht verändern. Um dies zu erreichen, müssen dem System derartige Transformationen bekannt sein.

Abbildung 4.7 zeigt ein komplexeres Beispiel, in dem sich Fall (obere Zeile) und Anfrage (untere Zeile) nur durch eine Transformation von Teilen unterscheiden. Bei beiden Layouts handelt es sich um eine Stammleitung und Verbindungsleitungen des Zuluftsystems. Die Vergleich könnte folgendermaßen ablaufen:

Die Anfrage wird achtfach repräsentiert, entsprechend den acht möglichen, die Nützlichkeit nicht beeinflussenden Transformationen. Sie bestehen aus Drehungen um Vielfache von 90° und einer möglichen Spiegelung.

Für jede Repräsentation wird das größte Matching gebildet. Es werden zwei größte Matchings gefunden (Spalte (b)), die drei Objekte gemeinsam beinhalten.

Diese beiden Matchings können kombiniert werden (Spalte (c)) und erlauben die Aussage, daß der linke Teil in Anfrage und Fall identisch ist und der rechte Teil der Anfrage im Fall gespiegelt vorkommt.

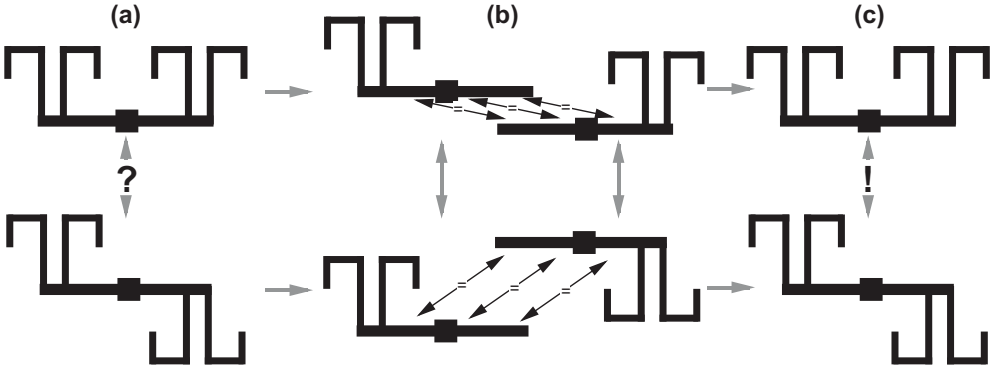

Abbildung 4.7: (a) zeigt zwei zu vergleichende Strukturen. Die Kenntnis der erlaubten Transformationen ermöglicht die Identifikation der gemeinsamen Teilstrukturen (b) und somit eine vollständige Zuordnung der Objekte beider Strukturen (c).

Durch die acht möglichen Transformationen vergrößert sich im allgemeinen die Anzahl der nicht erweiterbaren gemeinsamen Teilgraphen um das achtfache, zuzüglich der Kombinationen unterschiedlich transformierter Teilgraphen. Da ein vollständiges Verfahren alle diese Teilgraphen finden muß und die beschriebene Vorgehensweise dies genau macht, ohne daß neben der siebenfachen Transformation der Anfrage zusätzlicher Aufwand entsteht, ist der so beschriebene Vergleich optimal im Sinne der Effizienz. Ebenso wird kaum zusätzlicher Speicher benötigt, da der einzige Speichermehrbedarf durch die sieben Transformationen der Anfrage entsteht.

Es wäre bestimmt ebenso möglich, die Erkennung von transformierten Teilgraphen in den Vergleich zu integrieren, also direkt bei der Erweiterung jedes Teilgraphens die möglichen Transformationen zu beachten. Dieses könnte jedoch aufgrund der gezeigten Optimalität nicht effizienter sein und ist auf jeden Fall komplizierter zu realisieren.

4.2.2 Einführung komplexer Relationen

Wie in Abschnitt 4.1.6 diskutiert, hängt die Laufzeit eines Graphmatchings ungefähr von $N(k)^{clique_{max}}$ ab. $N(k)$ bezeichnet die durchschnittliche Anzahl der möglichen Erweiterungen eines Matchings und $clique_{max}$ die maximale Größe eines Matchings, also die maximale Anzahl der zuordbaren Relationen zweier Strukturen.

Die Idee der Definition komplexer Relationen basiert auf der Zusammenfassung mehrerer binärer Relationen zu wenigen mehrstelligen. Die mehrstelligen Relationen können auf die gleiche Weise in den Kompatibilitätsgraphen abgebildet werden wie die binären Relationen. Abbildung 4.8 zeigt ein Beispiel für mehrstellige Relationen. Es basiert auf primitiven Relationen, welche zweidimensionale Richtungswechsel (T-Kreuzung, X-Kreuzung oder L-Knick) und Parallelität beschreiben. Statt alle Beziehungen zwischen den linken Ellipsen zum Vergleich mit anderen Strukturen zu benutzen, wird die gesamte Struktur durch eine Reihe von Kreuzungen beschrieben. Würde man diese Struktur mit einer anderen Struktur, die auch eine Reihe von Kreuzungen enthält, vergleichen, so würden die Reihen aus Kreuzungen einander zugeordnet, und der Vergleich wäre in einem Schritt abgeschlossen. Bei der Benutzung des Ansatzes der komplexen Relationen für die in Kapitel 2 vorgestellten Relationen würden eine Reihe von Relationsknoten „Ellipse nahe bei Ellipse" durch einen Knoten ersetzt, der die mehrstellige Relation Reihe mit den senkrechten Ellipsen repräsentiert.

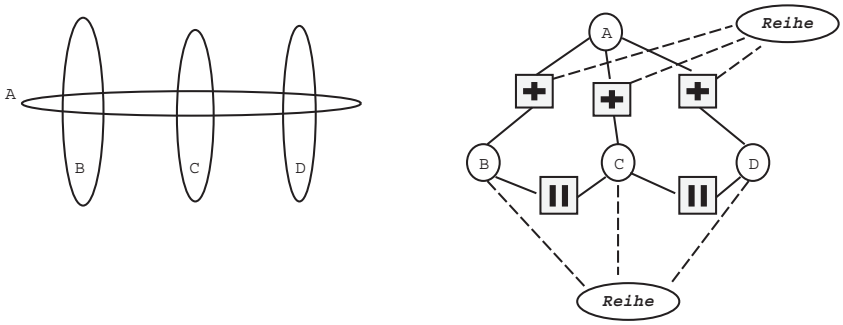

Abbildung 4.8: Um die Effizienz des Vergleichs zweier Strukturen zu steigern, werden mehrere binäre Relationen zu wenigen mehrstelligen zusammengefaßt.

Die komplexen Relationen werden nicht zufällig vorgegeben, sondern

repräsentieren Gestalten oder markante Teile von Gestalten. Gestalten sind Planfragmente, die von Benutzern als typische Teilstrukturen benannt werden. In [Schaaf, 1994] findet sich eine Diskussion der Bedeutung von Gestalten und auch eine Sammlung von Gestalten aus dem Bereich des orthogonalen Layouts. Zum Beispiel werden die Ellipsen aus Abbildung 4.8 als Fischgräte bezeichnet, welche sich häufig bei der Erschließung von Regionen finden, wie zum Beispiel bei der Verbindung von Lampen. Weitere Beispiele sind Kämme, Karrees, regelmäßig gefüllte Felder und andere mehr. Der Vergleich der beiden Konferenzsäle aus Abbildung 4.5 wäre nach der Erkennung solcher Gestalten zum Beispiel auch nach nur einem Schritt abgeschlossen, da die Stühle beider Säle ein regelmäßig gefülltes Feld bilden.

Die Verwendung komplexer Relationen ist ein vielversprechender Ansatz, um die Komplexität eines Strukturvergleiches zu senken. Es gibt jedoch noch mehrere offene Fragen, die zur Zeit im Rahmen einer Diplomarbeit bearbeitet werden und die in bisherigen Arbeiten noch nicht beantwortet wurden:

- Wie geht die Größe einer Gestalt in den Vergleich ein? Was bedeutet es zum Beispiel, daß eine Reihe von vier Objekten einer Reihe von sechs Objekten zugeordnet wird?
- Wie werden die komplexen Relationen beim Vergleich behandelt, falls sie nicht zugeordnet werden können? Eine Möglichkeit wäre zum Beispiel, eine erkannte Gestalt in binäre Relationen aufzulösen, falls die Gestalt als ganzes nicht Bestandteil des Matchings sein kann.
- Wie werden die Gestalten effizient in einer Struktur erkannt und verwaltet?
- Was passiert, wenn die Objektmengen zweier Gestalten sich überlappen?

4.2.3 Konsequenzen kontinuierlicher Kompatibilitätsfunktionen

Dieser Abschnitt diskutiert Kompatibilitätsfunktionen, die nicht auf {0,1} abbilden, sondern auf das Intervall [0,1]. Dies würde ermöglichen, auch unterschiedliche, aber ähnliche Relationen und Objekte einander zuzuordnen, falls es keine kompatiblen gibt.

Berührt zum Beispiel der Stuhl im Fall die Wand, steht jedoch in der Anfrage nur in der Nähe der Wand, so könnten die Stühle aus Anfrage und

Fall trotzdem einander zugeordnet werden. Dazu müßte man Ähnlichkeiten zwischen Relationen vorgeben. Genauso kann man auch eine Ähnlichkeit zwischen einem Stuhl und einem Sessel definieren und gegebenenfalls einen Stuhl einem Sessel zuordnen. Diese Ähnlichkeit könnte auch während der Benutzung des SFBS gelernt werden. Man könnte dem Benutzer erlauben, beliebige Objekte aus Fall und Anfrage einander manuell zuzuordnen und sich diese Zuordnungen merken.

Die Integration dieser Idee in das vorgestellten Konzept ist leicht möglich. Die Ähnlichkeit zweier dreidimensionaler Relationen würde sich aus den Ähnlichkeiten der eindimensionalen Relationen und den Ähnlichkeiten der beteiligten Objekte zusammensetzen. Man würde den Knoten des Kompatibilitätsgraphen Gewichte anfügen, die die Ähnlichkeit der Relationen beschreiben, deren Zuordnung durch den Knoten repräsentiert wird. Ein Knoten hätte ein Gewicht zwischen 0 und 1. Je höher sein Gewicht, desto ähnlicher die zugeordneten Relationen. Die größte Clique wäre nicht mehr definiert durch die größte Anzahl von Knoten, sondern durch die größte Summe der Gewichte der enthaltenen Knoten. Knoten mit Gewicht 0 brauchten nicht repräsentiert zu werden.

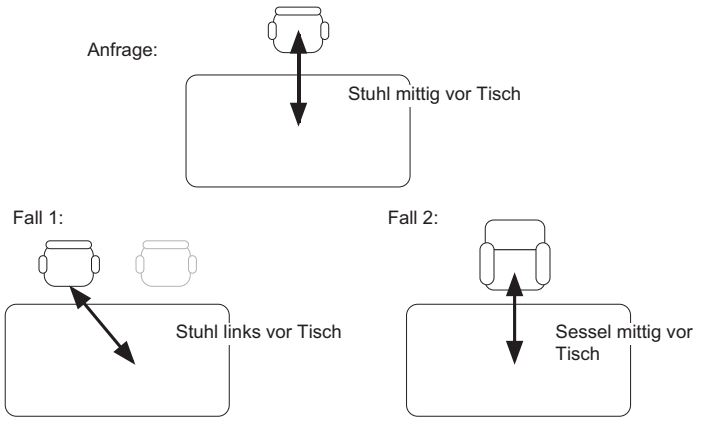

Abbildung 4.9: Was ist besser? Der Stuhl in ähnlicher Position wie in der Anfrage oder der ähnliche Sessel in identischer Position?

Ein mögliche Definition von Distanzen zwischen eindimensionalen Relationen zeigt die aus [Freksa, 1992a] entnommene Abbildung 4.10. Einige der Relationen sind durch Nachbarschaftskanten verbunden. Die Distanz zweier

Relationen berechnet sich aus der Anzahl der Kanten der kürzesten Verbindung. Seine 13 verschiedenen Relationen lassen sich direkt den von mir benutzten Relationen zuordnen, und die Distanz zweier dreidimensionaler Relationen läßt sich aus der Summe der Distanzen der eindimensionalen Relationen berechnen. Auf diese Weise ist eine Übertragung des Ansatzes von Freksa leicht möglich.

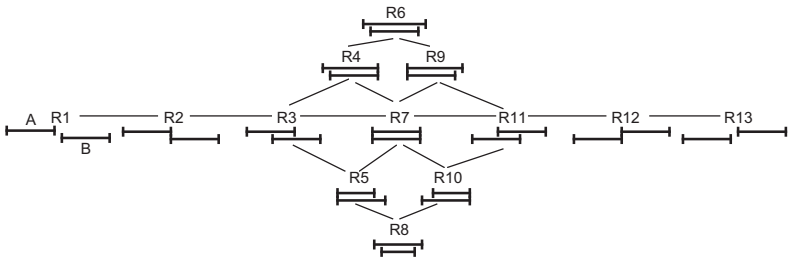

Abbildung 4.10: Die Distanz zweier Relationen berechnet sich aus der Anzahl der Kanten der kürzesten Verbindung. (Aus [Freksa, 1992a])

Prinzipiell sind also kontinuierliche Kompatibilitätsfunktionen kein Problem für das Konzept. In der Praxis führt diese Idee jedoch zu zwei von der vorgestellten Umsetzung des Graphvergleichs unabhängigen Problemen. Zum einen kann die Anzahl der möglichen Zuordnungen extrem ansteigen und somit die Antwortzeit des Vergleichs in die Höhe treiben. Zum anderen ist die Interpretation der Ähnlichkeiten zweifelhaft. Ist es zum Beispiel besser, wenn ein Matching die Zuordnung zweier ähnlicher Objekte mehr enthält und dafür eine Zuordnung zweier identischer weniger? Oder betrachte man zum Beispiel Abbildung 4.9. Ist das Matching mit Fall 1 besser, welches den Stuhl in ähnlicher relativer Postion enthält? Oder ist das Matching mit Fall 2 besser, weil der zum Stuhl ähnliche Sessel genauso vor dem Tisch steht wie der Stuhl in der Anfrage? Der vorgeschlagene Weg würde diese Fragen beantworten, aber es ist fraglich, ob die Antworten immer dem Wunsch des Benutzers entsprächen. Die Alternative, dem Benutzer alle möglichen Matchings anzuzeigen entfällt da es bei dem Vergleich größerer Situationen eine für den Benutzer unüberschaubare Menge von möglichen Matchings (fast) gleicher Bewertung geben kann.

4.3 Retrieval mit schnelleren Graphvergleichen

Ein Graphvergleich ist als Vergleichsfunktion für das Retrieval ungeeignet, da er im allgemeinen np-vollständig ist. Jede noch so gute Vorbereitung jeder noch so kleinen Fallbasis erfordert mindestens einen Aufruf der Vergleichsfunktionen, und somit besäße auch das Retrieval eine exponentielle Komplexität. Auf der anderen Seite hängt die Eignung eines Falles jedoch von dem Ergebnis des Graphvergleichs ab, Πund das Retrieval soll diejenigen Fälle finden, deren mit der Anfrage gemeinsamen Teilgraphen am größten sind. Die folgenden Abschnitte werden zunächst alternative Vergleichsfunktionen nennen, die in polynomieller Zeit Hinweise auf die Größe des maximalen gemeinsamen Teilgraphen liefern und zudem zeigen, wie man mit solchen abschätzenden Vergleichsfunktionen beim Retrieval optimal umgehen kann. Bisher gibt es kein Retrieval-Werkzeug, das Ähnlichkeitsmaße verarbeiten kann, die nur Ober- beziehungsweise Untergrenzen einer gesuchten Ähnlichkeit angeben.

4.3.1 Verschiedene Vergleichsfunktionen

Für den Vergleich von Fällen und Anfragen habe ich folgende Vergleichsfunktionen untersucht:

- Der Graphvergleich liefert die exakte Größe des gemeinsamen Teilgraphen, besitzt jedoch exponentielle Komplexität. Die Größe der Teilgraphen beschreibt die Anzahl der enthaltenen Knoten, also die Anzahl der im Matching enthaltenen Relationen. Eine Normierung der Ähnlichkeitsfunktion auf Werte zwischen 0 und 1, wie es von einigen Retrievalmechanismen gefordert wird, ist leicht möglich. Dazu wird die Größe des größten gemeinsamen Teilgraphen durch das Maximum der Größen der beiden verglichenen Graphen geteilt.

- Ein Mengenvergleich der enthaltenen Relationen, dividiert durch das Minimum der in Fall oder Anfrage vorhandenen Relationen, liefert in kurzer Zeit eine Obergrenze für die Größe des gemeinsamen Teilgraphen, da dieser nicht mehr Zuordnungen enthalten kann, als es kompatible Relationen in Anfrage und Fall gibt.

- Der Mengenvergleich der enthaltenen komplexen Relationen liefert zwar keine direkte Abschätzung des größten gemeinsamen Teilgraphen, jedoch ein gute Heuristik. Je mehr komplexe Relationen übereinstimmen, desto wahrscheinlicher ist ein großer gemeinsamer Teilgraph.

Man kann sich noch weitere Ansätze zur Ähnlichkeitsbestimmung zweier orthogonaler Layouts vorstellen. Diese besitzen jedoch im allgemeinen keinen direkten Bezug zur Größe des größten gemeinsamen Teilgraphens und liefern damit keine konkreten Hinweise auf die Anpassbarkeit eines Falles. Aus diesem Grund diskutiere ich sie nicht in dieser Arbeit. Ein interessanter Ansatz findet sich in [Coulon, 1993], welcher versucht die optische Ähnlichkeit zwischen Bauplänen zu bestimmen und dabei die Betrachtung aus verschiedenen Distanzen simuliert.

Bei den vom Retrieval durchgeführten Vergleichen muß man technisch zwei Arten unterscheiden: den direkten Vergleich einer Anfrage mit einem ausgewählten Fall und den Vergleich zweier Fälle zur Vorstrukturierung der Fallbasis. Die Vorstrukturierung der Fallbasis soll ermöglichen, durch einen direkten Vergleich eines Falles mit der Anfrage auch die Ähnlichkeiten der Nachbarn des direkt getesteten Falles einzuschätzen. Auf diese Weise können die zur Anfrage ähnlichsten Fälle gefunden werden, ohne vorher alle Fälle einzeln mit der Anfrage zu vergleichen.

Der direkte Vergleich ist sehr zeitkritisch, da lange Antwortzeiten das System unattraktiv machen (vergleiche Evaluationstabelle in Abbildung 6.21). Deshalb eignen sich zur Suche (vergleiche „search" in Abschnitt 3.3) in großen Fallbasen nur die polynomiellen Vergleichsfunktionen. Hat sich der Benutzer mit Hilfe dieser Vergleichsfunktionen eine kleine Menge von Fällen gewählt, so soll ein Retrieval-Werkzeug jedoch auch in der Lage sein, diese wenigen Fälle mit der Graphvergleichsfunktion exakt prüfen zu lassen (vergleiche „initial match" in Abschnitt 3.3).

Der Vergleich zweier Fälle zur Vorstrukturierung ist weniger zeitkritisch, da er vor dem Anfragezeitpunkt, zum Beispiel über Nacht, durchgeführt werden kann. Trotzdem kann man nicht alle Fälle exakt vergleichen, da schon ein einzelner Graphvergleich zweier Fälle wesentlich länger dauern kann als eine Nacht oder auch als einige Jahre. Wie in Abschnitt 6.6 diskutiert, sind derartig aufwendige Vergleiche nicht unwahrscheinlich.
Falls der Graphvergleich innnerhalb einer vorzugebenden Zeitschranke abgeschlossen ist, sollte er zur Vorstrukturierung benutzt werden. Überschreitet er diese Zeitschranke, so liefert der größte bisher gefundene Teilgraph eine Untergrenze für die maximale Größe. Die entsprechende Obergrenze erhält man durch den Mengenvergleich. Innerhalb des durch Ober- und Untergrenze definierten Intervalls liefert ein Mengenvergleich der komplexen Relationen Hinweise auf die wahrscheinliche Größe des maximalen gemeinsamen Teilgraphens. Die Frage ist nun, wie man mit Ober- und Untergrenzen umgeht und welches Retrieval-Werkzeug dafür benutzt werden kann.

4.3.2 Umgang mit Ober- und Untergrenzen

Die Schwierigkeit bei der Benutzung von abschätzenden Vergleichsfunktionen ist, daß bei der Einschätzung der Ähnlichkeit des Nachbarn eines Testfalles nicht immer die gleiche Vergleichsfunktion benutzt werden kann wie bei dem entsprechenden direkten Test. Will man zum Beispiel ausnutzen, daß ein zu einem unähnlichen Testfall T ähnlicher Nachbar N unähnlich zur Anfrage ist, so benutzt man implizit einmal die Ober- und einmal die Untergrenze. Es wird also zur Einschätzung der Nachbarn des direkt verglichenen Falles eine andere Vergleichsfunktion benutzt als zum direkten Vergleich.

Um dieses Beispiel zu erläutern, ist es einfacher, mit den Inversen der Ähnlichkeiten, den Distanzen zu rechnen. Die Distanz berechnet sich nach Normierung auf [0,1] aus der Differenz der maximalen Ähnlichkeit 1.0 und der Ähnlichkeit zweier Fälle. Zwei identische Fälle haben also die Distanz 0.0 und zwei maximal verschiedene die Distanz 1.0. Bei der Einschätzung der Distanzen der Nachbarn eines direkt getesteten Falles T wird ausschließlich die Dreiecksungleichung ausgenutzt. Abbildung 4.11 zeigt die drei Möglichkeiten, um auszunutzen, daß in dem Dreieck aus Anfrage A, Testfall T und Nachbar N je zwei Distanzen immer größer oder gleich der dritten sein müssen.

Die zweite Ungleichung beschreibt zum Beispiel den erwähnten Zusammenhang, daß ein zu einem zur Anfrage unähnlichen Testfall T ähnlicher Nachbar N unähnlich zu Anfrage ist. Mit Distanzen ausgedrückt heißt dies: Weil T weit entfernt von der Anfrage ist (große Distanz $\overline{AT}$) und N dicht bei T liegt (kleine Distanz $\overline{TN}$), muß auch N weit entfernt von A sein (große Distanz $\overline{AN}$). Kennt man nicht die exakten Distanzen, sondern nur Ober- und Untergrenzen, so lautet die entsprechende Aussage: Weil T mindestens U(AT) von A entfernt ist ($\overline{AT}$ über der Untergrenze U(AT) liegt) und N höchstens O(NT) von T entfernt ist, kennt man eine Untergrenze für die Distanz $\overline{AN}$ und zwar: $\overline{AN} \geq U(AT) - O(TN)$. In der Abbildung bedeutet dies, daß N außerhalb des mit II gekennzeichneten Kreises um A liegt.

Damit die erste Ungleichung gilt, muß man hingegen beide Male die Obergrenzen verwenden, also $\overline{AN} \leq O(AT) + O(TN)$. N muß innerhalb des mit I gekennzeichneten Kreises um A liegen. Die dritte Variante der Dreiecksungleichung liefert eine Abschätzung für $\overline{AN}$, falls $\overline{TN} \geq \overline{AT}$. Dann liegt N außerhalb des mit III gekennzeichneten Kreises um A.

Abbildung 4.12 zeigt zwei Extrembeispiele für die Benutzung der Dreiecksungleichung. Im linken Beispiel wird die Distanz $\overline{AN}$ auf Null eingeschränkt da A in T und T in N enthalten ist. Rechts wird die Distanz $\overline{AN}$

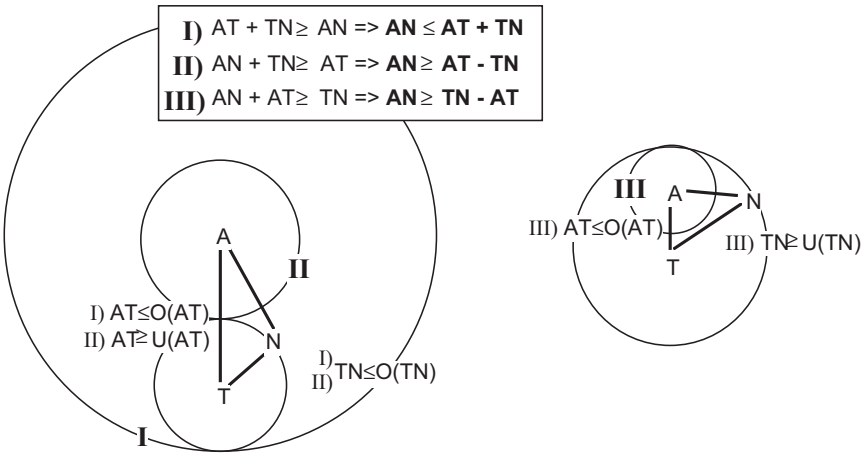

Abbildung 4.11: Bei einem direkten Vergleich eines Testfalles T mit der Anfrage A liefern alle drei Instanzen der Dreiecksungleichung zwischen A,T und N Einschränkungen für die Distanz des Nachbarfalles N zur Anfrage. Kennt man für die Distanzen $\overline{TN}$ und $\overline{AT}$ keine exakten Vergleichsfunktionen, so kann man, je nach gewählter Ungleichung, Ober- oder Untergrenzen benutzen.

Abbildung 4.12: Beispiele für die Benutzung der Dreiecksungleichung.

auf mindestens Eins geschätzt, da A in T enthalten ist, T und N aber disjunkt sind (Distanz 1.0).

Bisher gibt es noch kein Retrievalverfahren, welches alle drei Varianten der Dreiecksungleichung benutzt und mit Ober- und Untergrenzen umgehen kann. Die meisten Retrieval-Werkzeuge benutzten nur die erste Variante. Das Retrieval-Werkzeug ASPECT [Schaaf, 1996] ist jedoch gut für alle drei Varianten geeignet. Bisher sind zwar nur die ersten beiden Varianten vorgesehen, und es werden auch keine Unter- und Obergrenzen verarbeitet. Es ist jedoch das einzige Werkzeug, welches explizit die möglichen Distanzen zwischen Fällen und Anfragen einschränkt und zwischen den Vergleichsfunktionen für direkte und indirekte Vergleiche unterscheidet. Es kann somit leicht um die dritte Variante der Dreiecksungleichung und die Verarbeitung von Unter- und Obergrenzen erweitert werden.

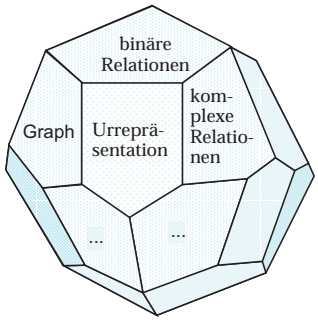

Abbildung 4.13: Jeder Fall enthält neben seiner Urrepräsentation alle für die verschiedenen Ähnlichkeitsmaße benötigten Repräsentationen aus [Schaaf, 1996].

Bei ASPECT stelle man sich die Fallbasis als einen Fallbasisraum vor. Jeder Fall dieses Raumes besitzt, zusätzlich zu seiner Urrepräsentation, eine Facette für jede von einem Ähnlichkeitsmaß benötigte Repräsentation, welche Aspekt genannt wird (vergleiche Abbildung 4.13). Zur Vorstrukturierung der Fallbasis wird die Ähnlichkeit zweier Fälle für jedes Aspektpaar mit dem entsprechenden Ähnlichkeitsmaß bestimmt und durch eine Kante zwischen den Aspekten repräsentiert (vergleiche Abbildung 4.14). Dadurch werden die Fälle in dem Fallbasisraum relativ zueinander positioniert. Die Idee ist, bei jedem Vergleich eines Falles mit der Anfrage die Ähnlichkeit seiner Nachbarn zur Anfrage abzuschätzen und als nächstes diejenigen Fälle direkt zu testen, die nach den bisherigen Abschätzungen noch in der Nähe der Anfrage liegen

können. Wird also zum Beispiel die Distanz $\overline{AT}$ abgeschätzt oder berechnet, so können für alle Nachbarn N von T die Distanzen $\overline{AN}$ entsprechend den drei Instanzen der Dreiecksungleichung abgeschätzt werden.

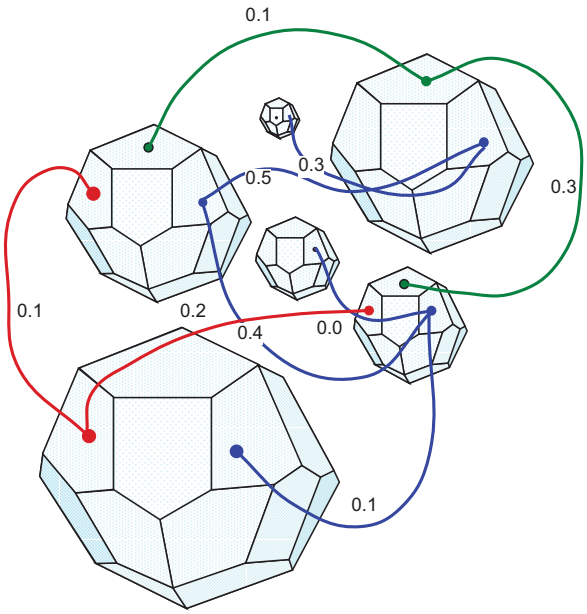

Abbildung 4.14: Besteht eine Ähnlichkeitsbeziehung zwischen Aspekten zweier Fälle, so wird sie durch ein Kante repräsentiert aus [Schaaf, 1996].

Als weitere Funktionalität kann ASPECT analysieren, wie oft und wie stark die Dreiecksungleichung für eine gegebene Ähnlichkeitsfunktion innerhalb einer gegebenen Fallbasis verletzt wird. Wird eine Verletzung festgestellt, so werden die auf dieser Ähnlichkeitsfunktion basierenden Abschätzungen entsprechend vorsichtig behandelt. Die Retrievalfunktionalität bleibt jedoch erhalten. Da der Graphvergleich im Prinzip eine Spezialform des Mengenvergleichs ist und dieser die Dreiecksungleichung im allgemeinen verletzt, hat ASPECT beim Test des Ähnlichkeitsmaßes erwartungsgemäß Verletzungen festgestellt. Diese Verletzungen waren jedoch in allen vorhanden Fallbasen so gering, daß ASPECT sie ausgleichen konnte.

Zusammenfassung der Antworten:

1. ⇒ Die Objekte und Relationen einer Struktur können die Knoten und Kanten eines Graphen bilden, wobei ich vorschlage, die Relationen als Knoten und die Objekte als Kanten zu verwenden. Graphen können mittels Graphmatching verglichen werden, wobei verschiedene Erweiterungen bekannter Verfahren die Effizienz steigern können. Das Ergebnis ist die größte gemeinsame Teilstruktur zweier Graphen.

2. ⇒ Die Einschränkung auf ausgewählte Zuordnungen (1-Kanten) reduziert die durchschnittliche Komplexität erheblich. Auch die anwendungsspezifische Verwendung komplexer Relationen kann zur Reduktion der Komplexität benutzt werden.

3. ⇒ Mengenvergleiche der Relationen zweier Graphen liefern Schranken zur Abschätzung der Größe des maximalen gemeinsamen Teilgraphens.

4. ⇒ Transformierte Teilmengen können durch mehrfachen Graphvergleich mit transformierten Anfragen gefunden werden. Die Komplexität der Suche ist dabei nicht größer, als wenn man die Suche nach Transformationen in den Graphvergleich einbaut.

5. ⇒ Durch kontinuierliche Kompatibilitätsfunktionen lassen sich nicht nur identische, sondern auch ähnliche Strukturen finden und verwenden. Derartige Kompatibilitätsfunktionen können problemlos in das vorgestellte Konzept integriert werden. Sie erhöhen jedoch die Komplexität und können zu fragwürdigen Ergebnissen führen.

6. ⇒ Die Verwendung aller drei Varianten der Dreiecksungleichung ermöglicht, die Nachbarn direkt getesteter Fälle maximal einzuschränken. Dies beschleunigt das Retrieval, da im Durchschnitt weniger Fälle direkt getestet werden müssen.

Kapitel 5

Anpassung

Fragen:

1. Wieso ist die Anpassung beim SFBS so einfach?
2. Wie können Übertragungsheuristiken gelernt werden?
3. Wie kann die Anpassung von Struktur und Objekteigenschaften kombiniert werden?
4. Wie benutzt man mehr als einen Fall zur Anpassung?

Die Anpassung benötigt den in Anfrage und Fall gemeinsamen Teilgraphen. Ist die Struktur des Falles beim Retrieval nicht mit der Struktur der Anfrage durch Graphmatching verglichen worden, so wird dieses vor Beginn der Anpassung durchgeführt. Dies kann zum Beispiel vorkommen, wenn der Benutzer einen Fall zur Anfrage ausgewählt hat, der nicht in der von „search" erzeugten Vorauswahl enthalten ist und somit nicht vom „initial match" verarbeitet wurde. Danach sind durch die Übertragungsheuristik und die im Fall erkannten Abhängigkeiten die zu übertragenden Teile des Falles und die zu beachtenden Abhängigkeiten bekannt. Deswegen ist im SFBS die Anpassung nur die Lösung eines bereits formulierten Constraint-Problems. Entsprechend kurz ist dies Kapitel. Das Constraint-Problem und seine Lösung werden in Abschnitt 5.2 beschrieben.

Die Anpassung basiert im SFBS auf der Menge der Zuordnungen (Matching) M, den vorgegebenen festen Objekten der Anfrage (O_{fest}), den freien Objekten der Anfrage (O_{frei}), den zur Übertragung ausgewählten Objekten des Falles (O_F) und den zur Übertragung ausgewählten Relationen des Falles (R_F) (vergleiche Abschnitt 3.3). Das Ergebnis ist die um die positionierten Objekte der Mengen O_{frei} und O_F ergänzte Menge O_{fest}. Für

alle Eingabemengen der Anpassung wurde erläutert, wie diese berechnet oder vom Benutzer vorgegeben werden können (vergleiche Abschnitt 3.3). Ein Nachteil bei der Auswahl der zu übertragenden Objekte ist, daß entweder relativ viele Objekte entsprechend der allgemeinen Übertragungsheuristik ausgewählt werden, oder aber man zusätzliches anwendungsspezifisches Wissen vorgeben muß.

Abschnitt 5.1 wird eine Idee beschrieben, wie gelernt werden kann, welche Objekte übertragen werden sollen. Der Anpassungsalgorithmus wird in Abschnitt 5.2 beschrieben und einige ausgewählte Aspekte des Algorithmus werden in den nachfolgenden Abschnitten diskutiert.

5.1 Eine Idee zum Lernen einer Übertragungsheuristik

Wie schon in Abschnitt 2.3.3 erläutert, sollen im allgemeinen die Objekte und Relationen aus einem Fall übertragen werden, die mit dem Matching in Beziehung stehen. Diese Heuristik bestimmt jedoch nur eine Obermenge der sinnvollen Übertragungen und ist entsprechend unpräzise.

Es gibt zwei Informationsquellen, aus denen man lernen kann, die Übertragungsheuristik zu präzisieren. Zum einen das Benutzerverhalten, zum anderen die Fallbasis. Falls Teile der Fallbasis nicht nur Layouts enthalten, die das Ergebnis zusammengesetzter Arbeitsschritte sind, sondern auch Ergebnisse einzelner Arbeitsschritte, kann aus den enthaltenen Objekten und Relationen gelernt werden, welche Typen von Objekten und Relationen semantisch zusammengehören. Die Übertragungsheuristik wird dann dahingehend eingeschränkt, daß nur diejenigen Typen von Objekten und Relationen übertragen werden, die semantisch zu den zugeordneten Objekten passen. Auf diese Einschränkung zielt auch das Lernen aus Benutzerverhalten.

Nicht nur Fälle liefern Informationen über semantische Zusammenhänge, sondern auch die zeitliche Abfolge der Benutzeraktionen. Objekttypen, die vom Benutzer in enger zeitlicher Abfolge erzeugt, verändert oder eingeblendet werden, liefern ebenso Hinweise auf einen semantischen Zusammenhang.

Eine automatische Einschränkung auf die Übertragung von Objekt- und Relationstypen, deren semantischer Zusammenhang mit dem Matching auf die beschriebene Weise ermittelt wurde, ist eine unter Umständen immer noch vage Heuristik. Deswegen sollte sie auch nur zur Vorauswahl der Übertragungsmenge benutzt und die endgültige Entscheidung dem Benutzer überlassen werden.

5.2 Der Anpassungsalgorithmus

Der Anpassungsalgorithmus soll ein Constraint-Problem lösen. Durch die Übertragung wird eine Anfragestruktur erzeugt, deren Relationen die Positionen von Objekten beschreiben. Die Bestimmung der absoluten Objektpositionen und die Auflösung möglicher Widersprüche bildet das durch die Anpassung zu lösende Constraint-Problem. Aufgrund von Unterschieden zwischen Anfrage und Fall wird das Constraint-Problem im allgemeinen überspezifiziert sein. Um es zu lösen könnte man beliebige relaxationsfähige Constraint-Satisfaction-Algorithmen benutzen. Diese besitzen jedoch exponentielle Komplexität. In diesem Abschnitt werde ich einen neuen linearen Anpassungsalgorithmus mit integrierter Relaxationsheuristik beschreiben, welcher schnell eine gute Lösung findet.

Im ersten Schritt der Anpassung (vergleiche Algorithmus in Abbildung 5.1) werden die Fallobjekte kopiert und der Menge O_{frei} angefügt. Die Transfer-Funktion überträgt im zweiten Schritt die Relation R_F auf die Anfrage. Vor dem Transfer besaß jede Relation Verweise auf die an der Relation beteiligten Fallobjekte. Diese Verweise werden durch den Transfer verändert. Verweise auf Fallobjekte aus M werden auf die zugeordneten Anfrageobjekte und die gerade erzeugten Kopien der Fallobjekte umgeleitet. Da die kopierten Fallobjekte frei sind und jede übertragene Relation mit mindestens einem dieser Objekte verbunden ist, ist mindestens eines der Objekte jeder Relation $R_i \in R_F$ frei. Mit dem zweiten Schritt ist die Übertragung abgeschlossen und die Anpassung kann begonnen werden.

Während der Anpassung wird die Menge der Relationen R_F abgearbeitet, bis sie leer ist und somit keine Relationen mehr rekonstruiert werden können (Schritt 3). In Schritt 3.2 werden diejenigen Relationen bestimmt, die in der aktuellen Iteration abgearbeitet werden sollen. Die Funktion $anwendbar(O_1, O_2, R)$ testet, ob die Position von O_2 durch die Relation R und die Position von O_1 festgelegt werden kann. Dazu muß die Position von O_1 „fest" ($O_1 \in O_{fest}$) und die von O_2 „frei" ($O_2 \in O_{frei}$) sein. Für n-stellige Relationen ($n > 2$) würde anwendbar(0_1, ...O_n,R_i) testen, ob mindestens ein Objekt „frei" und ein Objekt „fest" ist. Ist eine Relation anwendbar auf das k-te Objekt, so wird sie aus der Menge R_F entfernt und der Menge R_{Ak} angefügt. Ist eine Relation nicht anwendbar und sind alle beteiligten Objekte fest, so wird sie in 3.3.3 ebenfalls aus R_F entfernt. Die Relationen der Mengen R_{Ak} werden anschließend in Schritt 3.4 und 3.5 rekonstruiert, wodurch nach und nach alle Objekte fest werden, da die übertragene Struktur zusammenhängt. Sind alle Objekte fest, so wird R_F durch 3.3.3 geleert und

Anpassung($M, O_{fest}, O_{frei}, O_F, R_F$)

1. $O_{frei} := O_{frei} \cup$ kopiere(O_F)
2. $R_F :=$ transfer(R_F, M)
3. WHILE $R_F \neq \emptyset$
   - 3.1 $R_{A1} := \emptyset$, $R_{A2} := \emptyset$
   - 3.2 FOR $R_i \in R_F$ DO
   - 3.3 CASE
     - 3.3.1 anwendbar(objekt$_1(R_i)$, objekt$_2(R_i), R_i$) THEN $R_{A2} := R_{A2} \cup R_i$, $R_F := R_F \setminus R_i$
     - 3.3.2 anwendbar(objekt$_2(R_i)$, objekt$_1(R_i), R_i$) THEN $R_{A1} := R_{A1} \cup R_i$, $R_F := R_F \setminus R_i$
     - 3.3.3 objekt$_1(R_i) \in O_{fest} \wedge$ objekt$_2(R_i) \in O_{fest}$ THEN $R_F := R_F \setminus R_i$
   - 3.4 FOR $R_i \in R_{A2}$ DO
     - 3.4.1 obj$'_2$:=reconstruct(objekt$_1(R_i)$, objekt$_2(R_i), R_i$)
     - 3.4.2 $O_{frei} := O_{frei} \setminus$objekt$_1(R_i)$
     - 3.4.3 $O_{fest} := O_{fest} \cup$objekt$_1(R_i)$
   - 3.5 FOR $R_i \in R_{A1}$ DO
     - 3.5.1 obj$'_1$:=reconstruct(objekt$_2(R_i)$, objekt$_1(R_i), R_i$)
     - 3.5.2 $O_{frei} := O_{frei} \setminus$objekt$_2(R_i)$
     - 3.5.3 $O_{fest} := O_{fest} \cup$objekt$_2(R_i)$

Abbildung 5.1: Der Algorithmus zur Anpassung

der Algorithmus terminiert.

Da die von TOPO verwendeten Relationen unscharf sind, würden sie sich nicht eindeutig rekonstruieren lassen, wenn nicht jede Relation den zusätzlichen Parameter $\delta d/max(AB)$ enthalten würde (vergleiche Abschnitt 2.3.1). Dieser Parameter erlaubt auch bei unscharfen Relationen, wie zum Beispiel „A nahe bei B", die exakte Rekonstruktion der aus dem Fall übertragenen topologischen Relation. Nur falls die exakte Rekonstruktion zu einem Konflikt mit anderen Beziehungen führt, wird die Position des anzupassenden Objektes von der Rekonstruktionsfunktion, ausgehend von der exakt rekonstruierten Position und innerhalb des durch die Relation definierten Intervalls, möglichst geringfügig verändert. Ist eine Relation nicht rekonstruierbar, so wird sie durch Entfernung aus R_F vollständig relaxiert. Somit ist die Rekonstruktionsfunktion total definiert.

Der vorgestellte Algorithmus löst Constraint-Probleme durch eine eingebaute Relaxationsheuristik: Widersprechen sich zwei Constraints, so wird dasjenige vollständig relaxiert, das weiter von den festen Objekten des Matchings entfernt ist, da es später aus R_F zur Verarbeitung herausgenommen wird, als das nähere. „Weiter entfernt" bedeutet, daß die in der Anfrage zu rekonstruierende Relation im Fall nicht Objekte des Matchings betrifft, sondern Objekte die über einer Pfad aus Objekten und Relationen mit dem Matching verbunden ist. Die Länge des Pfades bestimmt die Entfernung.

Die Relaxationsheuristik ist bei der Anpassung durchaus sinnvoll, da die Entfernung zum Matching auch der Entfernung zur Problemstellung entspricht und somit Relationen, die direkt mit der Problemstellung zusammenhängen, Priorität vor anderen haben.

Der Algorithmus findet sehr schnell eine Lösung, da seine Laufzeit nur linear von der Länge n des längsten übertragenen Pfades aus n Relationen abhängt. In jeder Iteration (Schritt 3) wird ein weiteres Objekt dieses Pfades fest und somit sind nach n Iterationen alle Objekte dieses längsten Pfades, und somit auch aller kürzeren Pfade, fest. Bezüglich der Lösung kann zwar nicht garantiert werden, daß möglichst wenig Relationen vollständig relaxiert wurden, sie ist jedoch entsprechend der verwendeten Relaxationsheuristik eine gute Lösung.

Wollte man andere Relaxationsheuristiken oder -regeln benutzen, könnte man auch jeden anderen Constraint-Satisfaction-Algorithmus benutzen.

5.3 Konfliktlösung durch Benutzung von Statistiken

Einige Konflikte zwischen Relationen können auch dadurch gelöst werden, daß bei der Anpassung versucht wird, die Größe von Objekten anzupassen. Dies wurde konzeptionell in Abschnitt 3.5.2 und am Beispiel des zu schmalen Fensters in Abbildung 3.5 erklärt.

Betrifft eine Relation zwei feste Objekte, so läge nach der bisherigen Erklärung des Anpassungsalgorithmus ein Konflikt vor, und die Relation würde nie zur Rekonstruktion benutzt. Um derartige Konflikte, falls möglich, durch eine Anpassung der Objektgrößen lösen zu können, werden je Dimension die beiden Grenzen eines Objektes getrennt betrachtet. Die Rekonstruktionsfunktion der Relation „Wand A berührt Fenster B von rechts" fixiert zum Beispiel nur die linke Seite von Wand A und die rechte Seite von Fenster B. Falls nun zusätzlich gelten soll, daß Fenster B Wand C von rechts berührt, muß die Funktion *anwendbar* derart erweitert werden, daß $anwendbar(Fenster-B, Wand-C, R(B,C))$ prüft, ob das Fenster so verlängert oder verkürzt werden kann, daß die linke Grenze von Fenster B die Wand C berührt. Falls möglich, würde die nachfolgende Rekonstruktionsfunktion die Größenänderung durchführen. In TOPO sind entsprechende Funktionen für orthogonales Layout implementiert.

Diese Art der Konfliktlösung könnte auch mit beliebigen Constraint-Relaxations-Algorithmen erzielt werden. Man müßte nur das Constraint-Netz derart erweitern, daß die verschiedenen Grenzen eines Objektes als eigene Variablen repräsentiert werden und zwischen den Variablen eines Objektes Constraints definiert sind, welche die Größe des Objektes beschreiben. Die Relaxation dieser neuen Constraints entspricht der gezeigten Konfliktlösung durch Größenveränderung und könnte auch dieselbe Statistik benutzten.

5.4 Anpassung mehrerer Fälle

Im Gegensatz zum vorgestellten Konzept werden in anderen Ansätzen des fallbasierten Schließens mehrere Fälle zur Anpassung benutzt. In [Voß, 1996] findet sich eine Beschreibung solcher Ansätze. Ich habe mich gegen das gleichzeitige Anpassen mehrerer Fälle entschieden, weil es das Risiko birgt, daß sich zwei Fälle widersprechen. Um diese Risiko zu vermeiden, müßte Wissen bereitstehen, wie man eventuell auftretende Widersprüche löst. Al-

ternativ kann mit dem beschriebenen Konzept eine iterative Anpassung mehrerer Fälle durchgeführt werden, die gleichzeitig dazu führt, daß möglichst zueinander passende Fälle benutzt werden.

Soll zum Beispiel eine Menge von Fällen zur Anpassung benutzt werden, so wählt man den besten (vergleiche Abschnitt 3.4) Fall als ersten zur Anpassung aus und startet danach ein neues Retrieval, statt alle mit einem Retrieval auszuwählen. Der dann am besten bewertete Fall wird derjenige sein, der am besten zu der aus dem ersten Fall und der Anfrage entstandenen Ergebnis paßt. Nacheinander sucht und paßt man nun einen Fall nach dem anderen an. Können zwei Fälle nicht kombiniert werden, so wird die Distanz des zweiten Falles zum Ergebnis der Anpassung des ersten Falles auf die Anfrage größer sein, als zur ursprünglichen Anfrage.

Zusammenfassung der Antworten:

1. $\Rightarrow$ Im SFBS ist die Anpassung „relativ einfach", da die zu übertragenden Objekte und ihre Abhängigkeiten bekannt sind.
2. $\Rightarrow$ Aus Fällen und dem Benutzerverhalten kann eine Übertragungsheuristik gelernt werden.
3. $\Rightarrow$ Kennt man Abhängigkeiten zwischen Objekteigenschaften (gibt es zum Beispiel Statistiken für übliche Abstände zwischen beiden Rändern eines Objektes), so kann man die Anpassung der Objekteigenschaften in die Rekonstruktionsfunktionen integrieren.
4. $\Rightarrow$ Mehrere Fälle können nacheinander zur Anpassung benutzt werden.

Kapitel 6

Realisierung von TOPO

Fragen:

1. Wie sind die Daten (Objekte, Relationen, Fälle) und das Wissen repräsentiert?

2. Wie ist TOPO in das benutzte CAD-System integriert?

3. Wie sieht eine TOPO-Problemlösung ganz praktisch aus?

4. Wie kann der Benutzer mit der exponentiellen Komplexität umgehen?

Dieses Kapitel beschreibt die Realisierung von TOPO, des entsprechend dem vorgestellten Konzept des SFBS entwickelten Softwaremoduls. Neben den benutzten Datenstrukturen erläutert es insbesondere die Interaktion von TOPO mit dem CAD-System, dem im genannten FABEL-Projekt entwickelten Prototypen und dem Benutzer. TOPO wurde in Common Lisp [Steele, 1990] implementiert. Die ersten beiden Abschnitte dieses Kapitels beschreiben die zur Repräsentation von Objekten, Relationen, Anfragen, Fällen und Wissen benutzten Datenstrukturen. Da das Wissen auf die anderen Datenstrukturen angewendet wird, werden diese im ersten Abschnitt 6.1 beschrieben. Abschnitt 6.2 wird die das Wissen repräsentierenden Datenstrukturen zeigen. Die restlichen Abschnitte beschreiben die Integration von TOPO in das FABEL-System und einige Beispielszenarien, die die Benutzerinteraktion demonstrieren. Die Beispielszenarien sind dabei auf die Anpassung eines Falles beschränkt, der vorab von dem Retrieval-Werkzeug ASPECT[Schaaf, 1996] gefunden wurde (vergleiche Abschnitt 4.3).

6.1 Objekte, Relationen und Fälle

Die Implementierung des SFBS operiert auf Layouts, deren Objekte aus dem Komponentenmodell A4 [Hovestadt, 1994] stammen. A4-Objekte sind in Subsysteme wie Kaltwasser, Abluft, Fassade und viele andere unterteilt. Weitere Eigenschaften beschreiben ihre Geometrie, ihre abstrakte Funktion (Anschluß, Verbindung oder Erschließung), ihren Abstraktionsgrad (Elementierungsplanung, Linienplanung oder Bereichsplanung) und ihre Größenordnung (zum Beispiel Gebäude, Etage, Raum oder Teilbereich eines Raumes). Die Form der Objekte spielt keine Rolle, da im orthogonalen Raum alle Objekte durch rechteckige Hüllquader repräsentiert werden. Zur visuellen Verarbeitung sind sie um graphische Informationen, wie zweidimensionale Darstellung in verschiedenen Maßstäben, dreidimensionale Darstellung und weitere externe Graphikformate angereichert. Zur Unterscheidung besitzen sie außerdem eine eindeutige Identität. In TOPO wird für jedes Objekt des Layouts ein eigenes Objekt erzeugt, da das benutzte Graphiksystem unter dem Betriebssystem NeXT-Step arbeitet und TOPO unter UNIX.

Ein Objekt besitzt in TOPO alle semantischen und geometrischen Eigenschaften des A4-Objektes außer der Graphik (siehe Abbildung 6.1). Die semantischen Eigenschaften sind unter einem Typbezeichner `typ-b` zusammengefaßt. Zum Beispiel hat ein Zuluftverbindungsbereich der Größenordnung 6 den Typ „ZULVB6". Dieser Typ wird beim Vergleich zweier Objekte benutzt. Zusätzlich ist ein Eintrag vorhanden, der auf die Relationen verweist, an denen das Objekt beteiligt ist.

Topologische Relationen setzten sich in TOPO, wie in Kapitel 2 beschrieben, aus drei eindimensionalen Relationen zusammen. Jede eindimensionale Relation gehört zu einem der acht Typen aus Abbildung 2.6 und ist vom Typ `1-dim-rel`. Zusätzlich kennt eine dreidimensionale Relation ihre beiden Objekte und hat eine eindeutige Identität. Der Typbezeichner `typ-b` einer dreidimensionalen Relation setzt sich aus den drei Typen der drei eindimensionalen Relationen und den Typen der beteiligten Objekte zusammen. Zum Beispiel ist „ZULVB6-b0i<i<-ZULVB6" der Typ einer Relation „b0i<i<" zwischen zwei Zulufverbindungsbereichen „ZULVB6" sechster Ordnung. Der erste liegt in x-Dimension mit Abstand 0 links vom zweiten („b0") und umschließt ihn in den Dimensionen y und z links berührend („i<"). Das „links von" in horizontalen Dimensionen entspricht dem „unter" in der vertikalen.

Anfragen und Fälle werden bei TOPO durch die gleiche Datenstruktur repräsentiert (Abbildung 6.3). Der Eintrag `id` eines Falles repräsentiert

```
(defclass reduced-a4-object ()
 ((id      :type symbol  :initform 0   :accessor id  :initarg :id )
  (x       :type pos     :initform nil :accessor x   :initarg :x  )
  (dx      :type pos     :initform nil :accessor dx  :initarg :dx )
  (y       :type pos     :initform nil :accessor y   :initarg :y  )
  (dy      :type pos     :initform nil :accessor dy  :initarg :dy )
  (z       :type pos     :initform nil :accessor z   :initarg :z  )
  (dz      :type pos     :initform nil :accessor dz  :initarg :dz )
  (typ-b   :type string  :initform nil :accessor t-b :initarg :t-b)
  (rels    :type list    :initform nil :accessor rel :initarg :rel)
  ))
```

Abbildung 6.1: Die Datenstruktur eines A4-Objektes in TOPO enthält eine eindeutige `id`, die geometrische Position, einen Typbezeichner `typ-b` und eine Liste der Relationen, an denen es beteiligt ist. Diese Liste wird erst durch die Erkennungsfunktionen von TOPO bei der Verarbeitung eines Falles oder einer Anfrage gefüllt.

```
(defclass relation ()
  ((id      :type symbol            ...)
   (object1 :type reduced-a4-object ...)
   (object2 :type reduced-a4-object ...)
   (z-rel   :type 1-dim-rel         ...)
   (x-rel   :type 1-dim-rel         ...)
   (y-rel   :type 1-dim-rel         ...)
   (typ-b   :type string            ...)
   ))
```

Abbildung 6.2: Datenstruktur einer Relation

```
(defclass reduced-a4-case ()
 ((id          :type symbol  ...)
  (pathname    :type string  ...)
  (objects     :type list    ...)
  (object-hash :type t       ...)
  (abs-groups  :type list    ...)
  (rel-types   :type list    ...)
  (rel-groups  :type list    ...)
  (rel-gr-hash :type t       ...)
  ))
```

Abbildung 6.3: Neben einer `id` und einem Pfadnamen, die die Herkunft des Falles beschreiben, beinhaltet eine Fall seine Objekte und Relationen.

seine Position in der Datenbank des CAD-Systems. Sie besteht aus einem Projektnamen und einer weiteren ID, welche das Fallobjekt im CAD-System bezeichnet. Der Pfadname bezeichnet die Stelle, an der die Repräsentation des Falles gespeichert wurde. Die restlichen Einträge beschreiben die im Fall enthaltenen Objekte und Relationen. Die Hashtabellen (`object-hash` und `rel-gr-hash`) dienen dem schnellen Rekonstruieren von Verweisen nach dem Laden eines Falles. Dies ist nötig, da bei der Speicherung eines Objektes Verweise, wie zum Beispiel die Liste der Verweise auf Relationen, an denen das Objekt beteiligt ist, nur als Liste von ID's abgespeichert werden können.

Fälle bestehen in FABEL aus bis zu einigen Tausend Objekten. Es war daher in FABEL kein Problem, alle Objekte eines Falles in den Hauptspeicher zu laden. Ein Fall mit ungefähr 3000 Objekten besaß jedoch bis zu 400000 Relationen, welche in keinem unserer Computersysteme in den Hauptspeicher geladen werden konnten. Deshalb werden von TOPO die Relationen bei Bedarf nachgeladen. „Bei Bedarf" heißt für TOPO, wenn sie zum Graphvergleich oder zur Übertragung benötigt werden. Gegeben eine Anfrage und einen zu vergleichenden Fall, werden von TOPO alle Relationen geladen, die Objekte betreffen, welche den gleichen Typ wie die Objekte der Anfrage besitzen. Alle Relationen, die die gleichen Objekttypen betreffen, werden zu diesem Zweck in einer Datei abgespeichert (Abbildung 6.4). Die in einem Fall vorkommenden Paare von Typbezeichnern werden zum effizienten Zugriff zusätzlich im Slot `abs-groups` abgelegt.

Der Slot `rel-names` enthält eine Liste der Typbezeichner der enthaltenen Relationen, und `rel-groups` enthält einen strukturierten Verweis auf die

```
.../ca96-03-01-106/

ABLVH4-ABLVH5.rel       ABLVH5-ABLVH7.rel       ABLVH7-ABLVH8.rel
ABLVH4-ABLVH6.rel       ABLVH5-ABLVH8.rel       ABLVH8-ABLVH8.rel
ABLVH4-ABLVH7.rel       ABLVH6-ABLVH6.rel       case.topo
ABLVH4-ABLVH8.rel       ABLVH6-ABLVH7.rel       objects.topo
ABLVH5-ABLVH5.rel       ABLVH6-ABLVH8.rel
ABLVH7-ABLVH7.rel       ABLVH5-ABLVH6.rel
```

Abbildung 6.4: Verzeichnisstruktur eines gespeicherten Falles mit der ID „ca96-03-01-106". Das Verzeichnis enthält die gespeicherten Relationen, den gespeicherten Fall (case.topo) ohne Relationen und Objekte sowie die Objekte selber (objects.topo).

enthaltenen Relationen. Beide Slots werden im nächsten Abschnitt bei der Vergleichsfunktion genauer erklärt.

Würde TOPO für ein anderes Anwendungsgebiet als das des FABEL-Projektes eingesetzt, so könnte die Datenstruktur der Fälle beibehalten werden. Ebenso könnte die Implementierung des dynamischen Nachladens von Relationen übernommen werden, da sie unter Lisp mit beliebigen Objekttypen und Relationstypen umgehen kann.

6.2 Modelliertes Wissen

Entsprechend der Zerlegung in die drei einzelnen Dimensionen sind auch die Erkennungs- und Rekonstruktionsfunktionen für eindimensionale Relationen modelliert. Um eine dreidimensionale Relation zu rekonstruieren, werden alle eindimensionalen Relationen nacheinander rekonstruiert. Deswegen benötigt man auch nur acht Erkennungs- und Rekonstruktionsfunktionen für die 8^3 verschiedenen dreidimensionale Relationen.

Die acht Erkennungsfunktionen werden in der Funktion `compute-relation` (Abbildung 6.5) von der Funktion `comp-1-dim-rels` für jede Dimension aufgerufen. Um andere Erkennungsfunktionen zu benutzen, müßte die Funktion `comp-1-dim-rels` entsprechend geändert werden. Abschließend berechnet die Funktion `compute-type` den Typbezeichner.

Während des Retrievals und Graphvergleichs (vergleiche Abschnitt 4.3) wird durch die Vergleichsfunktion getestet, wieviele und welche Relationen aus Anfrage und Fall kompatibel sind. Der Vergleich bezieht sich da-

```
(defmethod compute-relation ((obj1 reduced-a4-object)
                             (obj2 reduced-a4-object))
  (let ((relation (make-instance 'relation :obj1 obj1 :obj2 obj2)))
    (comp-1-dim-rels relation)
    (setf (t-b relation) (compute-type relation))
    relation))
```

Abbildung 6.5: In der Funktion „compute-relation" wird die für zwei Objekte geltende Relation berechnet.

bei auf die Typbezeichner der Relationen. Um nicht jede Relation der Anfrage mit jeder Relation des Falles vergleichen zu müssen, beinhaltet der Slot `rel-names` einer Fallstruktur eine Liste der Typbezeichner der in der Fallstruktur enthaltenen Relationen. Um auch den Aufbau des Kombinationsgraphen (Abschnitt 4.1.2) effizient durchführen zu können, beinhaltet der Slot `rel-groups` Verweise auf alle enthaltenen Relationen, wobei alle Relationen mit gleichem Typbezeichner in einer Gruppe zusammengefaßt sind. Enthalten zum Beispiel Anfrage und Fall Relationen vom Typ „ZULVB6-b0i<i<-ZULVB6", so bildet das Kreuzprodukt der entsprechenden Relationsgruppen aus Anfrage und Fall die Knoten des Kombinationsgraphen.

Die generische Ähnlichkeitsfunktion wählt den Fall mit dem größten Matching (vergleiche Abschnitt 3.4), wobei der Benutzer statt dem automatisch gewählten Fall auch jeden beliebigen anderen zur Anpassung vorgeben kann.

Zur Auswahl der zu übertragenden Objekte und Relationen wird ausschließlich die generische Übertragungsheuristik verwendet. Alle Objekte eines Falles, die mit dem Matching über Relationen in Verbindung stehen, werden übertragen.

Für jede der acht verschiedenen Relationen gibt es in TOPO eine `reconstruct`-Funktionen, die von dem Anpassungsalgorithmus (Abbildung 5.1) aufgerufen wird. Die dortige `reconstruct`-Funktion für dreidimensionale Relationen ruft für jede Dimension die im folgenden beschriebenen `reconstruct`-Funktionen für eindimensionale Relationen auf. Für jede der acht möglichen Beziehungen ist eine Unterklasse der Klasse `1-dim-rel` definiert. In Abhängigkeit vom Typ der als Parameter übergebenen eindimensionalen Relation wird von Lisp diejenige `reconstruct`-Funktion aufgerufen, die für den Typ definiert wurde. Neben den beteiligten Objekten wird der Funktion auch die anzupassende Dimension übergeben. Das Beispiel zeigt die erste und letzte Zeile der Funktion für Relationen vom Typ `including`.

```
(defmethod reconstruct (obj1 obj2 (relation including) dimension)
  .
  .
  .
  obj2)
```

Abbildung 6.6: Je nachdem, welcher Typ von Relation übergeben wird, führt Lisp die entsprechende typgebundene Funktion aus. Das Beispiel zeigt den Kopf der Funktion für Relationen vom Typ „including". Das Ergebnis ist das zweite Objekte in angepaßter Position.

```
(defclass  pos ()
  ((value :type float  :initform 0   :accessor value :initarg :value)
   (state :type symbol :initform nil :accessor state :initarg :state)))
```

Abbildung 6.7: Jede Objektgrenze hat neben ihrem Wert ein Attribut „state", das bezeichnet, ob die entsprechende Grenze schon durch die Anpassung fixiert wurde.

Als Ergebnis wird das angepaßte Objekt `obj2` zurückgegeben.

Wie in Abschnitt 5.3 erwähnt, versuchen die `reconstruct`-Funktion, die Größen eines Objektes anzupassen, falls sich dadurch ein Konflikt zwischen zwei Relationen lösen läßt. Deswegen enthalten die Einträge in die Slots `x, dx, y, dy, z` und `dz` der Objekte Werte, die über ein zusätzliches Attribut `state` verfügen. Dieses bezeichnet den Zustand, den eine Grenze eines Objektes hat (vergleiche Abbildung 6.7). Der Zustand kann „fest" oder „frei" sein, so daß zum Beispiel einige Grenzen eines Objektes durch die Rekonstruktionen festgelegt sein können (vergleiche Abschnitt 5.2), während die anderen Objektgrenzen noch angepaßt werden können.

6.3 Integration in den FABEL-Prototypen

Der FABEL-Prototyp dient der fallbasierten Unterstützung von CAD-Zeichnern. Er ist von 18 verschiedenen Entwicklern auf zwei verschiedenen Betriebssystemen in zwei verschiedenen Entwicklungsumgebungen entwickelt worden. Dies lag daran, daß jedes Betriebssystem und jede Sprache verschiedene Stärken und Schwächen hat. Die CAD-Umgebung M5PSEdit und die CAD-Datenbank A5Broker sind unter NeXTStep von der Universität

Karlsruhe entwickelt worden, da die dortigen Entwickler das Graphiksystem von NeXTStep zum damaligen Zeitpunkt für einzigartig hielten. Die von den anderen Projektpartnern entwickelten Werkzeuge sind in LISP unter UNIX entwickelt worden, um die hervorragenden Prototyping-Eigenschaften von LISP und den vorhandenen Maschinenpark zu nutzen. Beide Umgebungen sind über eine TCP/IP-Schnittstelle miteinander verbunden (siehe Abbildung 6.8). Zusätzlich existiert eine Anbindung an das kommerzielle CAD-System MiniCAD©[1], um Daten mit projektfremden Anwendern auszutauschen.

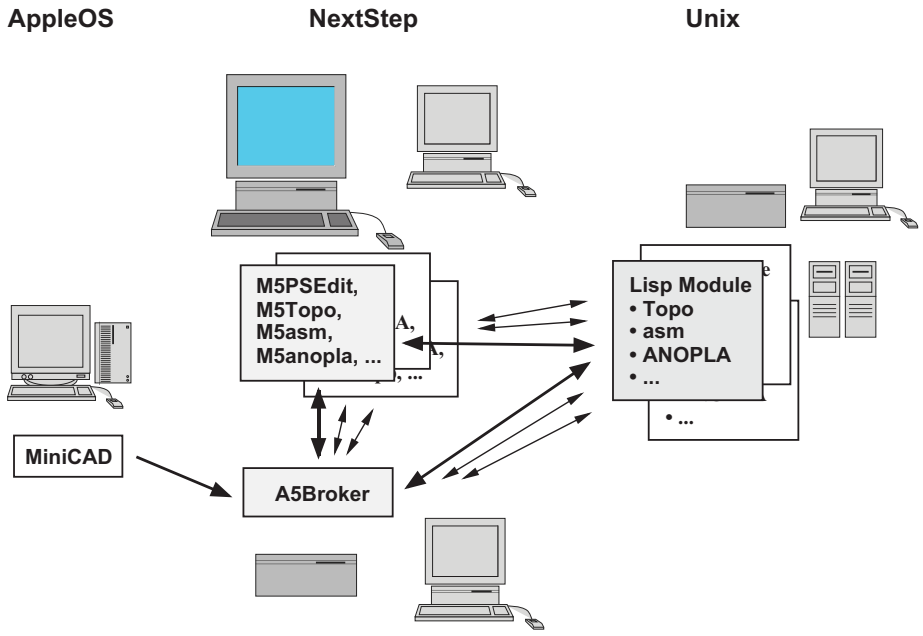

Abbildung 6.8: Vernetzung der Module des FABEL-Prototypen.

Neben dem CAD-System M5PSEdit gibt es für jedes betriebssystemfremde Werkzeug ein Modul unter NeXTStep, das die Kommunikation zwischen Werkzeug und Benutzer übernimmt. Sein Name besteht dabei aus der Abkürzung M5, als Bezeichner für ein Modul der fünften Version des CAD-Werkzeugs, und dem Werkzeugnamen. Also zum Beispiel „M5Topo"

[1] MiniCAD© ist unter MacOS das am weitesten verbreitete CAD-System.

für TOPO.

Ein Teil des Interfaces, das sogenannte „naive Benutzerinterface", ist in M5PSEdit integriert. Es ist jedoch kein funktionaler Bestandteil von M5PSEdit, sondern als funktionales Grafikobjekt Bestandteil der Daten. Es besitzt eine räumliche Position, den durch einen Rahmen präsentierten sogenannten Arbeitsbereich, und eine Auswahl von Bedienungselementen. Über die Bedienelemente können Funktionen ausgelöst werden, die die vom Benutzer innerhalb des Arbeitsbereiches markierten Objekte verarbeiten. Durch diese räumliche Positionierung von Werkzeugen entsteht eine Umgebung, die in FABEL als „virtuelle Baustelle" bezeichnet wird. Diese Metapher soll ausdrücken, daß verschiedene Architekten oder Bauingenieure bei der Planerstellung auf ähnliche Weise zusammenarbeiten können wie Arbeiter auf einer Baustelle: Sieht man sich einen Teil der Baustelle an, zum Beispiel den Deckenhohlraum einer Etage, so erkennt man anhand der da liegenden Werkzeuge, ob dort schon jemand anderes arbeitet.

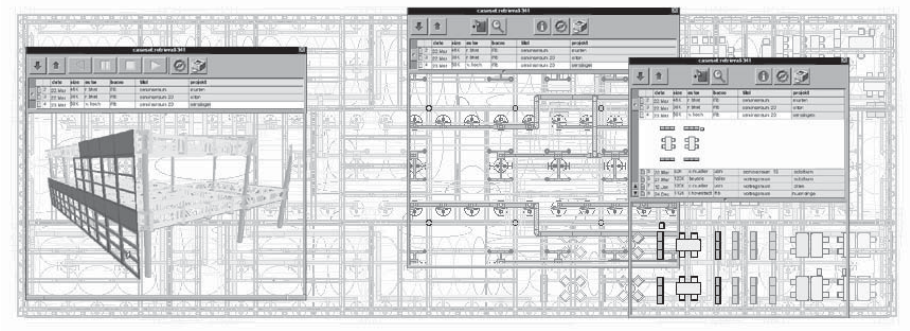

Abbildung 6.9: Ein Ausschnitt einer virtuellen Baustelle.

Abbildung 6.9 zeigt einen Ausschnitt einer „virtuellen Baustelle". Auf dem im Hintergrund liegenden Plan eines Geschosses sieht man drei Werkzeuge. Es sind ein 3-D-Betrachtungswerkzeug (links) und zwei Browser, die das Ergebnis zweier Retrievalanfragen zeigen.

6.4 Szenario

Dieser Abschnitt beschreibt anhand eines Beispiels die Bearbeitung einer Anfrage durch TOPO. Zuerst werden einige vorbereitende Schritte durch-

geführt und dann wird die Anfrage selektiert. Abschnitt 6.4.3 beschreibt die Benutzerinteraktion während der Anpassung.

6.4.1 Vorbereitende Schritte

Bei Aktivierung des TOPO-Interfaces erscheint ein sogenannter Inspektor, mit dessen Hilfe alle über die Möglichkeiten des naiven Benutzerinterfaces hinausgehenenden Interaktionen gesteuert werden. Der erste Schritt ist die Initialisierung der TCP/IP-Verbindung durch Angabe des Rechnernamens (inklusive der Portnummer) des Computers, auf welchem das TOPO-Werkzeug unter LISP gestartet wurde (Abbildung 6.10). Der Inhalt des Inspektors ersetzt sich nach der Initialisierung automatisch durch das für den nächsten Schritt zuständige Fenster.

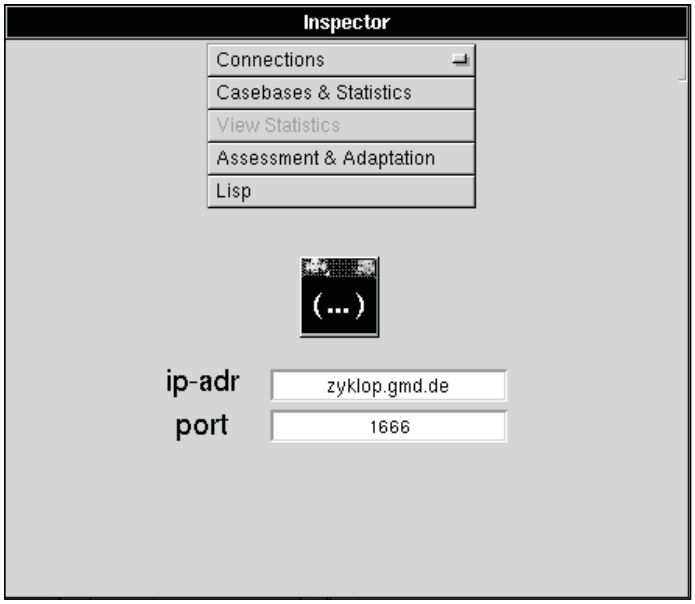

Abbildung 6.10: Initialisierung der TCP/IP-Verbindung.

Im nächsten Schritt wird die Statistik ausgewählt, die TOPO zur Anpassung zur Verfügung stehen soll. Dazu kann man entweder erst eine Fallbasis auswählen und eine Statistik erzeugen lassen oder direkt eine gespeicherte Statistik laden (Abbildung 6.11).

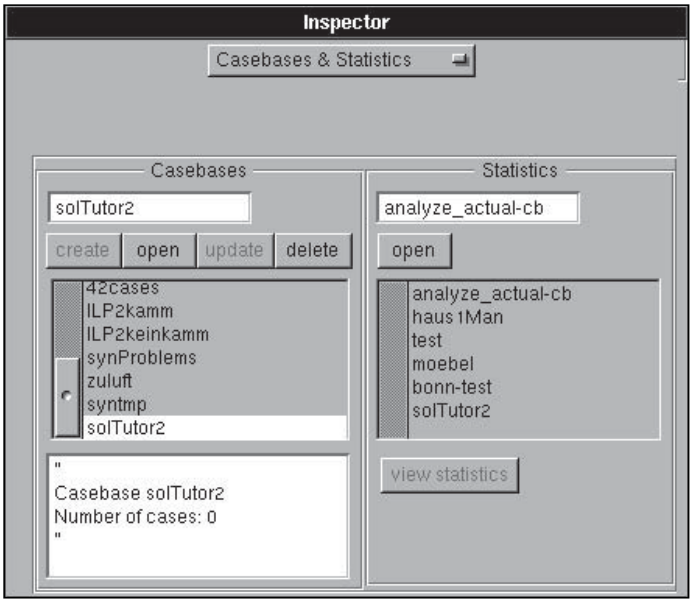

Abbildung 6.11: Auswahl der Fallbasis und der Statistik.

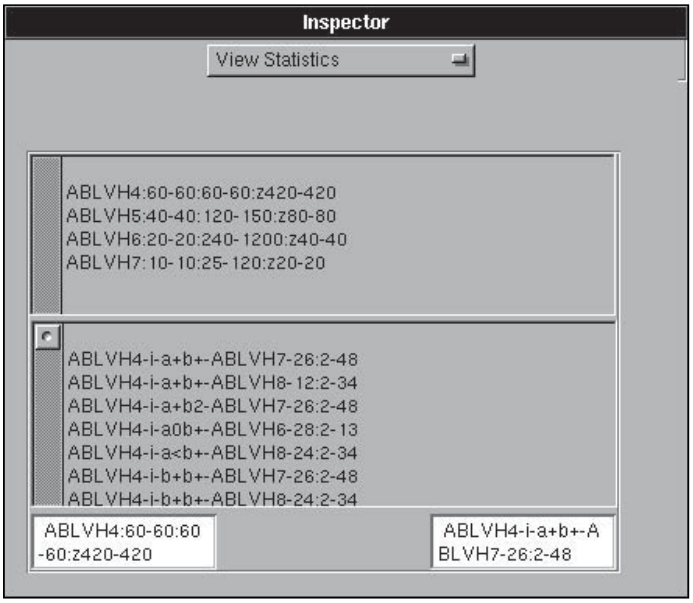

Abbildung 6.12: Anzeige der Objekt und Relationsstatistiken.

Abbildung 6.12 zeigt den Zustand des Inspektors, nachdem der Benutzer den Unterpunkt „View Statistics" aus dem POP-UP-Menu ausgewählt hat. Gezeigt sind die in der Fallbasis vorkommenden Objektgrößen und relativen Relationshäufigkeiten. Für die Objektgrößen werden im oberen Teil Intervalle der vorkommenden maximalen horizontalen, der minimalen horizontalen und der vertikalen Maße angegeben. Die horizontalen Ausdehnungen werden nicht entsprechend der Dimensionen x und y, sondern entsprechend der Minima und Maxima angegeben, da die Objekte horizontal in Schritten von 90° drehbar sind. Die zweite Zeile von oben bedeutet zum Beispiel, daß die vorkommenden Objekte des Typs ABLVH5 40cm breit, zwischen 1.20m und 1.60 lang und 80cm hoch sind.

Der untere Teile beschreibt die Häufigkeiten der Relationen und die Häufigkeiten der beteiligten Objekte. Auf diese Weise kann der Benutzer einschätzen, wie üblich eine Relation für eine Fallbasis ist. Die oberste Zeile bedeutet zum Beispiel, daß die Relation „ABLVH4-i-a<b<-ABLVH7" 26-mal vorkommt und es zwei Objekte vom Typ ABLVH4 und 48 Objekte vom Typ ABLVH7 gibt. Damit ist sie für ABLVH4 sehr häufig und üblich für ABLVH7.

6.4.2 Auswahl der Aufgabe

Abbildung 6.13 zeigt eine Anpassungsaufgabe für TOPO. Rechts sieht man die Anfrage, links einen mit Hilfe des TOPO-Ähnlichkeitsmaßes (vergleiche Abschnitt 4.3) gefundenen Fall. Der Benutzer selektiert alle Objekte und klickt auf den Analyseknopf des naiven Interfaces. Daraufhin berechnet TOPO die in Anfrage und Fall vorkommenden Relationen, und der Inhalt des Inspektors wechselt, um die erkannten Relationen zusammen mit ihren relativen Häufigkeiten zu zeigen (Abbildung 6.14).

Erkennt der Benutzer, daß eine Relation nicht nur unüblich, sondern auch falsch ist, so kann er die Position der an der Relation beteiligten Objekte freigeben. Dazu wählt er die Relation aus und markiert eines oder beide der beteiligten Objekte als „frei", indem er den unter dem Objektnamen befindlichen Knopf „Correct object" betätigt. Der Effekt einer solchen Aktion wird an einem weiteren Beispiel in Abschnitt 6.5 demonstriert. Um die Auswahl der Problemstellung abzuschließen, aktiviert der Benutzer das Graphmatching durch den „Match"-Knopf des naiven Interfaces.

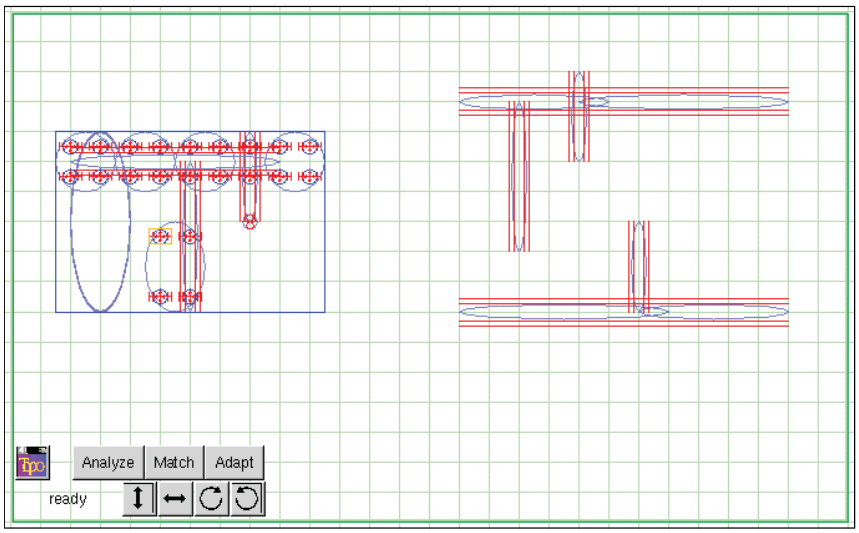

Abbildung 6.13: Rechts sieht man die Anfrage, links einen dazu passenden Fall und links darunter das naive Interface von TOPO.

6.4.3 Auswahl des Matchings und der zu übertragenden Objekte

Da das Graphmatching np-vollständig ist, kann die Antwortzeit des Matchingprozesses extrem lang sein. Um dem Benutzer die Möglichkeit zu geben einzugreifen, zeigt ein zusätzlicher Prozeß Informationen über den aktuellen Zustand des Matchingprozesses an und ermöglicht, ihn vorzeitig zu beenden (Abbildung 6.15).

Der Überwachungsprozeß ist zur gleichen LISP-Umgebung verbunden und kann deswegen direkt in den Matchingprozeß eingreifen. Bricht der Benutzer vorzeitig ab, so bilden die bisher gefundenen Matchings das Ergebnis. Um dem Benutzer eine Entscheidungsgrundlage zu bieten, kann er sich zwei Informationen anzeigen lassen. Die „Quality" beschreibt das Größenverhältnis zwischen größtem gefundenen Matching und dem größtmöglichem. „Progress" zeigt, wieviel Prozent des Suchraumes bisher abgesucht wurden.

Die gefundenen Matchings werden im Inspektor (Abbildung 6.14) unten links zur Auswahl aufgelistet. Standardmäßig wird das größte gewählt. Wurde der Matchingprozeß vorzeitig unterbrochen und der Benutzer ist mit

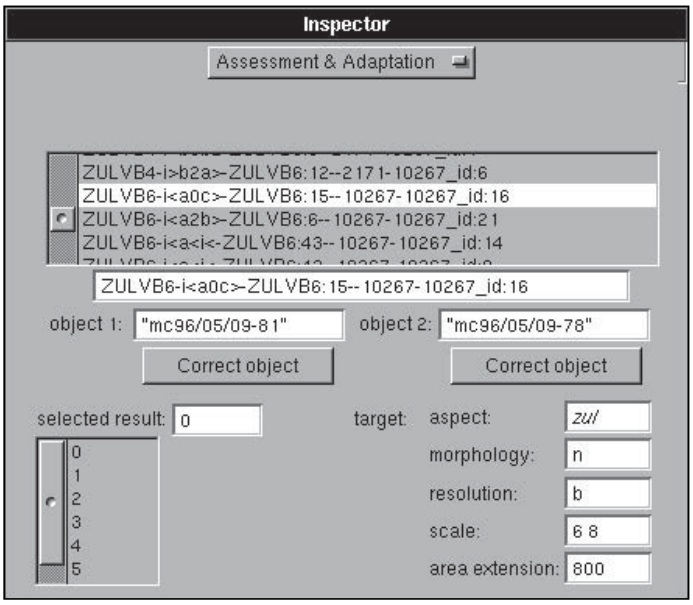

Abbildung 6.14: Der obere Teil des Inspektors zeigt die erkannten Relationen und ihre relativen Häufigkeiten. Unten links werden nach dem Matching die verschiedenen Matchings aufgelistet, und unten rechts kann der Benutzer einstellen, welche Objekte übertragen werden sollen.

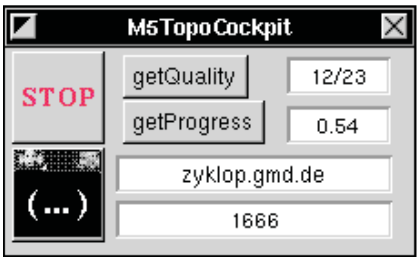

Abbildung 6.15: Der Prozeß „M5TopoCockpit" überwacht den Matchingprozeß und bietet die Möglichkeit, ihn vorzeitig zu beenden.

keinem der Matchingergebnisse zufrieden, so kann er das Problem vereinfachen. Dazu wählt er nur den Teil aus Anfrage und Fall aus, dessen Vergleich er für wesentlich hält.

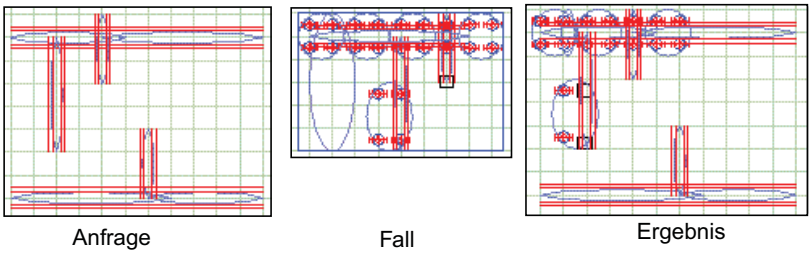

Abbildung 6.16: Das Ergebnis einer Anpassung mit TOPO.

Der nächste Schritt ist die Übertragung von Objekten und Relationen auf die Anfrage (vergleiche Kapitel 5 und Abschnitt 3.5). Entsprechend der allgemeinen Übertragungsheuristik (vergleiche Abschnitt 2.3.3) wird alles aus dem Fall übertragen, was zum gewählten Matching in Verbindung steht. Um die Menge der zu übertragenden Objekte einzuschränken, kann der Benutzer unten rechts die zu übertragenden Objekttypen vorgeben. Zusätzlich kann er die maximale Entfernung eines Objektes zum Matching angeben, in der ein Objekt übertragen werden soll.

Abbildung 6.16 zeigt das Ergebnis der Anpassung mit TOPO. Entsprechend der Vorgabe des Benutzers (siehe Abbildung 6.14 unten rechts) wurden die Zuluftnutzungsbereiche sechster und achter Ordnung auf die Anfrage übertragen. Die gezeigte Ergänzung einer Anfrage scheint ziemlich einfach zu sein. Im Detail erfordert sie jedoch genausoviel Aufwand wie die im nächsten Abschnitt gezeigten, scheinbar komplexeren Beispiele. Die Funktionalität wird die gleiche bleiben:

- Durch das Matching wird festgestellt, welcher Teil des Falles welchem Teil der Anfrage entspricht.

- Die Anpassung rekonstruiert die Position jedes einzelnen übertragenen Objektes durch Rekonstruktion seiner Beziehungen zu den Objekten des Matchings.

6.5 Weitere Beispiele

Dieser Abschnitt zeigt zwei Beispiele zur Korrektur, Größenanpassung und die Benutzung ganzer Gebäudepläne zur Anpassung.

Das erste Beispiel (Abbildung 6.17) zeigt den gleichen Fall wie im vorhergehenden Beispiel und fast die gleiche Anfrage. Die neue Anfrage enthält zusätzlich Zuluftschächte, aber der senkrechte Verbindungsbereich oberhalb des linken Schachtes ist kürzer als der des Falles.

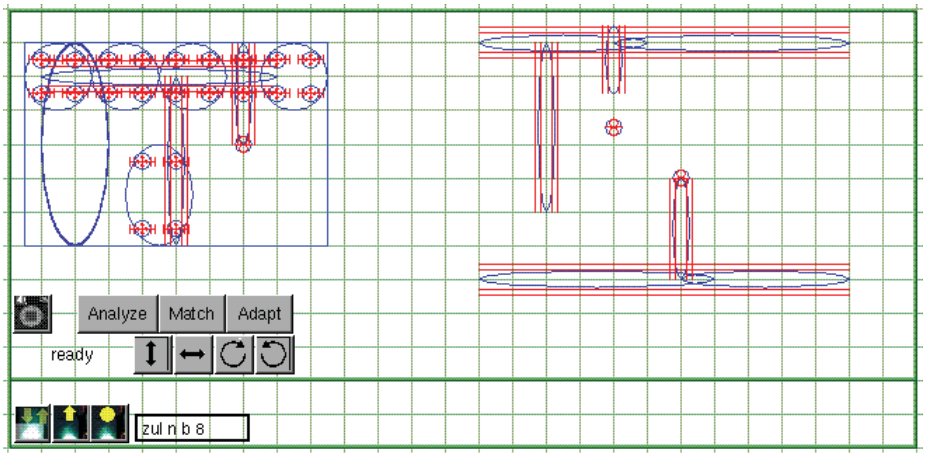

Abbildung 6.17: Im Unterschied zu Abbildung 6.16 enthält die hier gezeigte Anfrage Zuluftschächte (mit zwei kleineren Kreisen gefüllte Kreise), und die Zuluftverbindungsbereiche unterscheiden sich teilweise zu denen des Falles.

Die Analyse der Relationen erkennt, daß die Relation zwischen diesem kürzeren Verbindungsbereich und dem Schacht unüblich ist und zeigt dies im Inspektor an (Abbildung 6.18): $***$ZULVB6-i<a2a<-ZULVB6:0−−10267−10267$***$−id:22. Diese Zeile sagt aus, daß die Relation 22 vom Typ ZULVB6-i<a2a< −ZULVB6 0-mal in der Fallbasis vorkommt, obwohl die beteiligten Objekte je 10267-mal vorhanden sind. Dies wird durch die Sternchen hervorgehoben. Insgesamt wurden zwischen den 32 Objekten des Falles 292 Relationen erkannt und 22 Relationen zwischen den 9 Objekten der Anfrage.

Der Benutzer markiert die Position des Verbindungsbereiches als „frei". Dazu wählt er die unübliche Relation aus und betätigt den „Correct object"-

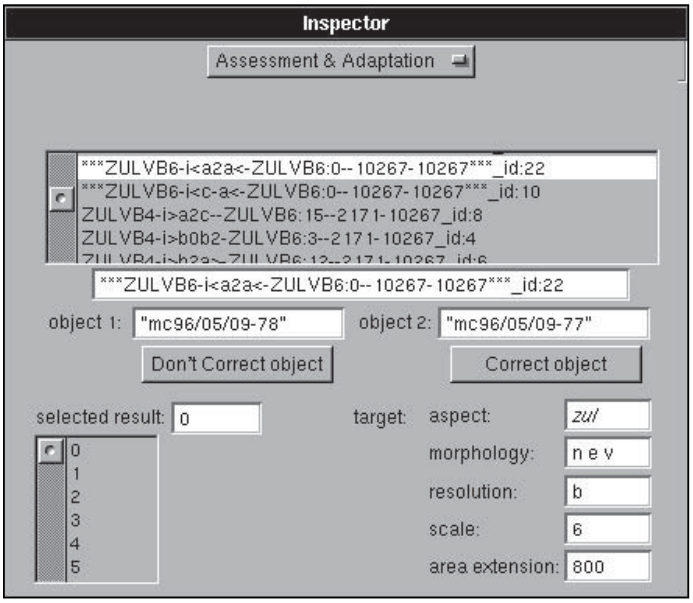

Abbildung 6.18: Im oberen Teil des Inspektors werden die erkannten Relationen angezeigt. Unübliche sind durch Sternchen markiert.

Knopf für den Verbindungsbereich (ID: mc96/05/09-78). Der Knopf wechselt daraufhin seine Funktion. Nochmaliges Betätigen würde die Freigabe rückgängig machen („Don't correct object"). Das nachfolgende Matching dauert ungefähr 200ms.

Diesmal werden im Beispiel die Zuluftnutzungsbereiche achter Ordnung nicht zur Übertragung ausgewählt (Inspektor Abbildung 6.18 rechts unten), um im Ergebnis der Anpassung (Abbildung 6.19) die Details besser erkennen zu können.

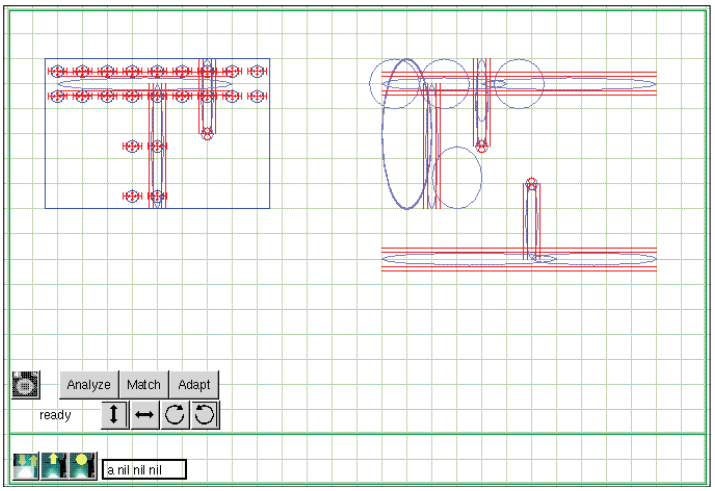

Abbildung 6.19: TOPO zeigt eine Alternative zur Position des freien Objektes an (der senkrechte Verbindungsbereich der von oben den oberen Schacht überlappt.

Alternativ zur alten Position des als „frei" markierten Verbindungsbereiches hat TOPO eine Kopie mit neuer Position erzeugt. Dabei hat sich dessen Größe verändert.

Wie in Abschnitt 3.4.2 begründet, hat TOPO den „freien" Verbindungsbereich der Anfrage dem rechten senkrechten Verbindungsbereich des Falles zugeordnet. Dieser steht im Fall zu denjenigen Objekten in Beziehung, die den mit dem „freien" Verbindungsbereich in Beziehung stehenden Objekten der Anfrage im Matching zugeordnet wurden.

Obwohl das Fallobjekt als Vorlage für die neue Position des Verbindungsbereiches benutzt wurde, ist der neue Verbindungsbereich länger als seine Vorlage im Fall. Dies liegt daran, daß auch die Relationen des Fallob-

jektes übertragen wurden und der Verbindungsbereich im Fall den Schacht überschneidet. Die Rekonstruktion dieser Beziehung hat in der Anfrage den Verbindungsbereich entsprechend verlängert, nachdem die neue Objektgröße mit der Statistik der üblichen Objektgrößen überprüft wurde.

Das zweite Beispiel zeigt die Benutzung eines ganzen Gebäudeplanes als Fall (Abbildung 6.20). Die Anfrage (a) besteht aus einem Zuluftschacht (links unten), zwei Astleitungen, einem Erschließungsbereich (Ellipse) und einem unüblich plazierten Winkel aus Zweigleitungen. Zwischen den Objekten erkennt TOPO elf Relationen.

Der Fall besteht aus ungefähr 3.500 Objekten und 395.000 Relationen und zeigt die Subsysteme eines ganzen Gebäudes. Während es bei den bisherigen Beispielen eventuell aufgrund der geringen Größe der Fälle so aussah, als ob die gezeigten Anpassungsaufgaben einfach sein könnten, haben manche Betrachter den hier gezeigten Fall aufgrund seiner Größe als unbrauchbaren „Fliegendreck" bezeichnet. Für TOPO bleibt die Aufgabe jedoch die gleiche und sogar die Antwortzeit bleibt ungefähr gleich.

Im ersten Schritt wird der Fall mit der Anfrage verglichen. Das Matching, ohne Prozeßkommunikation und Nachladen von Relationen des Falles, dauert ungefähr fünf Sekunden auf einer Sparc20. Das liegt daran, daß nur ein Bruchteil der Relationen des Falles den gleichen Typbezeichner wie die Relationen der Anfrage haben. Nur diese werden nachgeladen und den Relationen der Anfrage zugeordnet. Bei diesem Beispiel greift auch die Verbesserung des Matching-Verfahrens (vergleiche 4.1.4) besonders stark, da es im Gebäude viele Zuluftobjekte gibt, aber die meisten nicht in Beziehung zueinander stehen. Ohne diese Effizienzsteigerung lag die Antwortzeit bei über 30 Stunden.

TOPO bietet zwei mögliche Alternativen zu dem falsch plazierten Zweigleitungswinkel an, die TOPO in dem Fall gefunden hat ((c) und (d)). Der Benutzer entscheidet sich für (c) (e zeigt das Resultat der Anpassung) und wählt, ohne vorher noch einmal das Matching aktivieren zu müssen, einige Objekte des Falles zur Ergänzung zwecks Detaillierung (f) und Erweiterung (g) aus. Die Erweiterung (g) mag übertrieben und unbegründet sein, zeigt aber beispielhaft den möglichen Informationsreichtum eines Falles und damit die Mächtigkeit des fallbasierten Schließens.

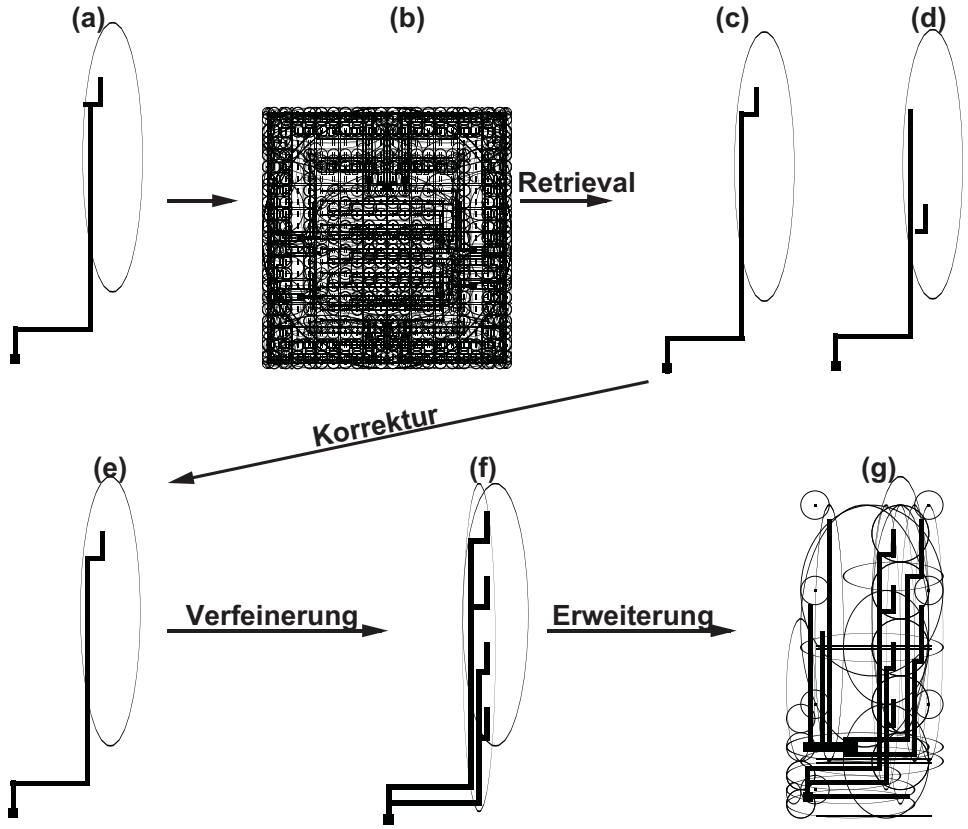

Abbildung 6.20: TOPO kann auch ganze Gebäudepläne (b) als Fall benutzen.

Bsp.	Relationen			0-	1-	Laufzeit	Laufzeit
	Anfrage	Fall	Knoten	Kanten	Kanten	BK	BK+
1	12	342	5	2	8	16ms	16ms
2	12	342	41	32	172	700ms	200ms
2a	12	12	24	135	79	2.03s	183ms
3	9	395.000	687	4579	612	30h	5.34s

Abbildung 6.21: Diese Tabelle zeigt einige Beispiele für das Laufzeitverhalten von max-clique$_{BK}$ und max-clique$_{BK+}$. Beispiel 1 bezieht sich auf Fall und Anfrage aus Abbildung 6.16, Beispiel 2 auf Abbildung 6.17, Beispiel 2a auf den Vergleich der Anfrage aus Abbildung 6.17 mit sich selbst und Beispiel 3 auf Abbildung 6.20.

6.6 Praktische Evaluation

Eine praktische Evaluation der Effizienz von TOPO kann nur oberflächlich und eine praktische Evaluation der Qualität nur subjektiv sein. Denn bezüglich der Effizienz schwanken die Antwortzeiten von TOPO stark in Abhängigkeit von der genauen Beschaffenheit der zu vergleichenden Graphen. Bezüglich der Qualität würde eine Befragung von Architekten viele verschiedene unterschiedliche subjektive Beurteilungen liefern [Coulon und Gebhardt, 1994]. Trotzdem möchte ich an dieser Stelle einige Beobachtungen zusammenfassen.

Die Laufzeit von TOPO steigt exponentiell mit der Anzahl der möglichen Zuordnungen. Eine Präsentation der mittleren Laufzeiten in Abhängigkeit von der Anzahl der möglichen Zuordnungen macht keinen Sinn, da diese stark, im Bereich zwischen einigen Millisekunden und mehreren Jahren schwankt. Um trotzdem die Benutzung des Verfahrens zu ermöglichen, habe ich den Kontrollprozeß zur Einschätzung des bisherigen Matchings und zur vorzeitigen Unterbrechung implementiert. Hat TOPO nach einigen Sekunden erst einen Bruchteil des Suchraumes abgesucht, so sollte der Benutzer das Problem neu formulieren.

Abbildung 6.21 gibt Information über den Vergleich der drei bisher gezeigten Beispiele. Die Anzahl der Knoten des Kombinationsgraphen entspricht der Anzahl der möglichen Zuordnungen von Relationen der Anfrage zu Relationen des Falles. Da die meisten Relationen inkompatibel sind, ist die Knotenzahl sehr klein. Dies führt zu entsprechend kurzen Laufzeiten. Während sich beim ersten Beispiel die Unterscheidung der 1-Kanten von den 0-Kanten nicht meßbar auf die Laufzeit auswirkt, wird der Unterschied der Laufzeit zwischen max-clique$_{BK}$ und max-clique$_{BK+}$ in den restlichen

Beispielen deutlich sichtbar.

Die Laufzeit bleibt auch bei anderen Beispielen klein (unter einer Sekunde), solange es keine großen Gruppen (>20) von räumlich benachbarten kompatiblen Objekten in Fall *und* Anfrage gibt. Besonders Reihen von Objekten gleichen Typbezeichners führen zu überraschend hohen Antwortzeiten, was durch die Verwendung komplexer Relationen vermieden werden könnte (vergleiche Abschnitt 4.2.2). Zerlegt der Benutzer seine Anfrage in entsprechend kleine Teilprobleme, so wird TOPO nach einigen Iterationen von jeweils wenigen Sekunden Laufzeit einen beliebigen gewählten Fall auf eine Anfrage angepaßt haben.

Die Qualität der Lösung hängt maßgeblich vom Fall ab, da für TOPO nur primitive Relationen modelliert und implementiert wurden. Jeder mit dem von TOPO zur Verfügung gestellten Ähnlichkeitsmaß gefundene Fall ist anpaßbar, jedoch bewertet TOPO nicht die Qualität des Falles. Genauere Betrachtungen zur Bewertung der Ergebnisse eines Retrievals findet man in [Coulon und Gebhardt, 1994].

Die praktische Benutzbarkeit von TOPO konnte nicht ausreichend evaluiert werden, da daß verbundene CAD-System ein Forschungsprototyp ist und nur vom kleinen Kreis der Entwickler intensiv benutzt wird. Deren Erfahrungen mit TOPO werden im folgenden kurz zusammmengefaßt. Da die Entwickler zwar Architekten sind, jedoch weitreichende Erfahrungen mit computergestütztem Arbeiten besitzen, sind ihre Anmerkungen eher als punktuelle Beurteilungen zu werten denn als repräsentative Erkenntnisse:

- Bei der Übertragung räumlicher Strukturen aus einem Fall auf die Anfrage durch Benutzung von TOPO bleiben häufig alle wichtigen Eigenschaften des Falles erhalten. Dies gilt unabhängig vom bearbeiteten Teilgebiet des Anwendungsbereiches.

- Probleme treten vor allem durch zwei Faktoren auf, welche schon in Abschnitt 3.5.2 konzeptionell diskutiert wurden:

  - Können Teile der Anfrage keinem Teil des Falles durch den Matchingprozeß zugeordnet werden, so kann es zu Konflikten zwischen diesen nicht zugeordneten Teilen und den aus dem Fall übertragenen Teilen kommen. Darauf wählen Benutzer nach kurzer Zeit bevorzugt Fälle zur Anpassung aus, die die Anfrage umschließen. Dieses Verhalten sollte in die Definition zukünftiger Ähnlichkeitsfunktionen einfließen (diskutiert in Abschnitt 3.4.1).

  - Da die benutzten Relationen unscharf sind, kommt es vor, daß sich Relationen der übertragenen Fallstruktur nicht rekonstruieren lassen. Weil TOPO kein Wissen über die Bedeutung verschiedener Relationen zur Verfügung steht, kann zufällig die Rekonstruktion unwichtiger Relationen der Rekonstruktion wichtiger Relationen vorgezogen werden. Dieses ist auf den ersten Blick für Benutzer unverständlich und irritierend. Deshalb wäre es wünschenswert, Wissen über die Bedeutung von Relationen zu aquirieren und bei der Anpassungen die wichtigen Relationen zuerst zu rekonstruieren.

- Meldete M5TopoCockpit Vergleiche hoher Komplexität, so brachen die Benutzer sofort ab. Mit der Zeit bewährte es sich, die Anfrage so exakt wie möglich auszuwählen. Der Vergleich war nicht nur schneller, sondern das Ergebnis auch präziser.

- Die statistische Überprüfung der Relationen findet häufig viele unübliche Relationen, von den nur wenige durch wirkliche Konflikte hervorgerufen werden. Dies liegt an der relativ schwachen Überdeckung möglicher Anfragen durch die analysierten Fälle der Fallbasis.

- Das Vorgeben der zu übertragenden Objekttypen schränkt die zu übertragenden Teile präzise auf die Benutzerwünsche ein und erwies sich als intuitiv besonders gut zu bedienende Funktionalität.

- Trotz der auftretenden Probleme waren die Benutzer erstaunt und erfreut von der praktischen Einsetzbarkeit von TOPO. Dies lag vor allen Dingen daran, daß TOPO immer funktionerte – unabhängig von den gerade bearbeiteten Objekttypen, ob nun Tisch, Wand, Zuluftleitung oder Trägerelement. Daß das Ergebnis aufgrund des fehlenden Wissens mal besser und mal schlechter war, kompensierten die Benutzer durch manuelle Veränderung der Ergebnisse oder erneute Anfrage. Der Aufwand war jedoch immer geringer, als bei vollständiger manueller Bearbeitung.

Dieses Kapitel enthält weniger allgemein verwendbare Antworten, als vielmehr eine Veranschaulichung der Realisierung von TOPO und dessen Nützlichkeit. Die wichtigste Eigenschaft läßt sich folgendermaßen zusammenfassen:

⇒ *Die Realisierung von TOPO übernimmt für den Benutzer große Teile der Anpassung und spart dem Benutzer somit Zeit und Aufwand. Der Problemlöseprozeß ist transparent und ermöglicht dem Benutzer steuernd einzugreifen. Dadurch kann auch unter anderem das benötigte Graphmatching exponentieller Komplexität benutzt werden, ohne die Kontrolle dem Benutzer durch unerträglich hohe Antwortzeiten zu entziehen.*

Kapitel 7

Spezialisierung und Generalisierung des Modells

Fragen:

1. *Für welche Anwendungsgebiete eignet sich SFBS?*
2. *Welchen Zusammenhang gibt es zwischen der zur Verfügung stehenden Fallbasis und der zu erwartenden Qualität des Ergebnisses?*
3. *Welche Teile des Modells von TOPO sind in wieweit generisch?*
4. *Was kann man an dem Modellierung von TOPO verfeinern, um in Teilgebieten bessere Ergebnisse zu erzielen?*
5. *Welche anderen Anwendungsgebiete kann man sich noch vorstellen?*

Dieses Kapitel zeigt Perspektiven, das in Kapitel 2 beschriebene und für TOPO benutzte Modell zu verändern, um andere Anwendungsgebiete zu erschließen. Prinzipiell eignet sich jedes Anwendungsgebiet, für das eine Fallbasis vorhanden oder erzeugbar ist und das benötigte anwendungsspezifische Wissen modelliert werden kann. Abschnitt 7.1 diskutiert wünschenswerte Eigenschaften einer Fallbasis.

Manche Anwendungsgebiete kommen neben der Fallbasis völlig ohne anwendungsspezifisches Wissen aus (Abschnitt 7.4.1). Das anwendungsspezifische Wissen, welches am Beispiel des orthogonalen Layouts modelliert wurde, kann verallgemeinert, verfeinert oder vollständig ersetzt werden. Abbildung 7.1 verdeutlicht die Möglichkeiten.

Zum einen kann man TOPO derart verallgemeinern, daß es als generisches Werkzeug zum strukturorientierten fallbasierten Schließen verwendet werden kann. Diese Möglichkeit wird Abschnitt 7.2 beschrieben. Abschnitt 7.3 beschäftigt sich mit der Verfeinerung des Modells, um in Teilgebieten des orthogonalen Layouts Ergebnisse höherer Qualität zu erzielen. Der letzte Abschnitt zeigt das für alternative Anwendungsgebiete notwendige Modell am Beispiel nicht-orthogonaler Layouts in einer GIS Anwendung und dem Vergleich funktionaler Gebäudespezifikationen.

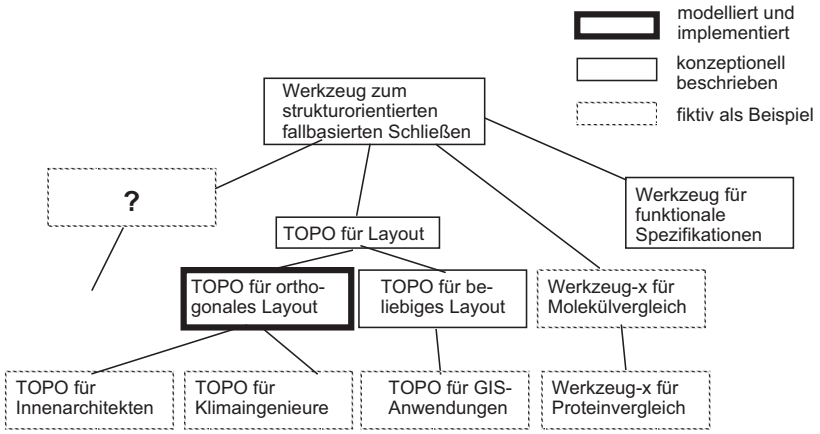

Abbildung 7.1: Verschiedene Instanziierungen des Konzeptes zum strukturorientierten fallbasierten Schließen.

7.1 Die Fallbasis

Die beiden wichtigsten Eigenschaften einer Fallbasis sind die Qualität der enthaltenen Fälle und ihre zu erwartende Nützlichkeit für zukünftige Anfragen. Die Qualität der enthaltenen Fälle bestimmt die Qualität dessen, was auf eine Anfrage übertragen wird und somit die Qualität des Ergebnisses. Die Nützlichkeit wird durch die Ähnlichkeitsfunktionen abgeschätzt. Ähnlichkeit von Fällen zu zukünftigen Anfragen ermöglicht die Übertragung der in den Fällen enthaltenen Eigenschaften, ohne viel an dem, was übertragen wird, zu verändern. Unter diesen Aspekt fällt auch die Größe einer Fallbasis. Weitere Eigenschaften sind, insbesondere in Hinblick auf die Eignung eines

Anwendungsgebietes, die Granularität und Aktualität der Fälle. Alle diese Eigenschaften werden im folgenden diskutiert.

Die Qualität einzelner Fälle läßt sich selten numerisch bewerten. Insgesamt sollten alle Fälle Lösungen oder Situationen beschreiben, die vom Benutzer akzeptiert werden. Idealer Weise sind alle Fälle vom Experten kommentiert, so daß ein späterer Benutzer auf Besonderheiten hingewiesen wird. Auf diese Weise könnten auch Negativbeispiele verwendet werden. Besitzt eine Anfrage Ähnlichkeit zu einem Negativbeispiel, so liefert der Kommentar Hinweise darauf, ob auch die Anfrage die gleichen negativen Eigenschaften besitzt. Für eine Anpassung sind Negativbeispiele jedoch nicht zu verwenden, da man keine negativen Eigenschaften auf die Anfrage übertragen möchte, und somit leisten sie keinen konstruktiven Beitrag. Zusätzlich müßte darauf geachtet werden, daß die Eigenschaften der Negativbeispiele nicht in die Statistiken der üblichen, und damit positiven, Eigenschaften und Relationen aufgenommen werden. Statt dessen könnten die Eigenschaften der Negativbeispiele in eine eigene Statistik aufgenommen werden, um nicht nur unübliche, sondern auch negative Eigenschaften in Anfragen zu erkennen.

Je größer die Unterschiede zwischen dem in Bezug auf die Anfrage bestmöglichen Fall und der Anfrage sind, desto mehr muß das, was auf die Anfrage übertragen wird, angepaßt werden. Die positiven Eigenschaften des Falles können dabei verloren gehen. Je dichter jedoch die Fälle die im Anwendungsgebiet erwarteten Anfragen überdecken, desto geringer sind die zu erwartenden Unterschiede. Unter diesem Gesichtspunkt ist eine relativ zur Größe des Anwendungsgebietes möglichst umfangreiche und gleichverteilte Fallbasis wünschenswert.

In einigen Anwendungsgebieten sind Fallbasen ein natürliches Nebenprodukt der täglichen Arbeit. Beispiele dafür sind Anwendungen, in denen einzelne Arbeitsschritte zwecks Abrechnung oder späterer Verwendung dokumentiert werden. Dies ist zum Beispiel in Arztpraxen oder Werkstätten der Fall. Die alten Fälle bestehen aus Patientenakten oder Reparaturprotokollen. Sie enthalten in der Regel kleine, abgeschlossene, semantisch zusammenhängende Einheiten, da die Fälle einzelne Arbeitsschritte repräsentieren oder diese in den Fällen erkennbar sind. Dies wird im folgenden unter einer hohen Granularität der Fälle verstanden. Ist eine Anfrage zu einem solchen Fall ähnlich, so kann alle Information aus dem Fall aufgrund des semantischen Zusammenhangs auf die Anfrage übertragen werden. In anderen Anwendungsgebieten, wie zum Beispiel der Architektur oder Stadtplanung, werden die einzelnen Arbeitsschritte nicht so isoliert dokumentiert, sondern es entsteht ein immer vollständigeres Layout, in dem Ausgangssi-

tuationen und Ergebnisse aller Arbeitsschritte miteinander verwoben sind. Da die enthaltenen Arbeitsschritte zum Teil voneinander abhängen, sind sie semantisch stark miteinander vernetzt. Solche Fälle niedriger Granularität erfordern Heuristiken oder anwendungsspezifisches Wissen, um zu entscheiden, was aus einem Fall auf die Anfrage übertragen werden soll (vergleiche Abschnitt 2.3.3).

Ein Problem, das im Laufe der Benutzung eines fallbasierten Systemes auftreten wird, ist die Wartung der Fallbasis. Es muß darauf geachtet werden, daß die alten Fälle aktuell bleiben, also Veränderungen im Anwendungsgebiet in die Fälle übertragen werden. Enthalten zum Beispiel Layoutfälle Bauteile, die nicht mehr lieferbar sind oder die mittlerweile durch bessere oder preiswertere ersetzt werden können, so müssen die alten Fälle aktualisiert oder gelöscht werden. Dieses Problem tritt zum Beispiel bei der fallbasierten Konfiguration von Telekommunikationsanlagen [Emde, 1996] auf. Dort ist eine interessante und einfach zu übertragende Lösung gewählt worden. Es werden nur abstrakte Komponenten benutzt, die mit Hilfe einer stets aktualisierten Komponentenliste im Anfragefall konkretisiert werden.

Zusätzlich müssen die Veränderungen auch in die Statistiken eingearbeitet werden, die die üblichen Objekteigenschaften repräsentieren. Diese Aktualisierung kann bei Bekanntwerden einer Veränderung im voraus durchgeführt werden oder in dem Moment, in dem ein Fall zur Anpassung ausgewählt wurde.

7.2 Generalisierung des Modells

Während die Erkennungs- und Rekonstruktionsfunktionen des in Kapitel 2 beschriebenen Modells sich nur zur Verarbeitung orthogonaler Layouts eignen, gelten andere Teile des Modells und der Implementierung auch für andere Anwendungsgebiete. Die Vergleichsfunktion für Objekte kann auch für nicht-orthogonales Layout verwendet werden, da sie die Geometrie nicht betrachtet. Noch allgemeiner ist die Vergleichsfunktion für Relationen. Unabhängig von der Art der Relation prüft sie auf Identität, benutzt dabei die Vergleichsfunktion für Objekte und ist somit unabhängig vom Anwendungsgebiet einsetzbar. Die Übertragungsheuristik (vergleiche Abschnitt 2.3.3) wählt diejenigen Objekte zur Übertragung aus, die mit dem gemeinsamen Teilgraphen über Relationen in Verbindung stehen. Da die Relationen anwendungsspezifisch vorgegeben werden, kann man die Übertragungsheuristik ebenfalls überall anwenden.

Auch die in Kapitel 3 benutzten Heuristiken sind generisch. Zur Auswahl des Falles und des Matchings wird die Größe des größten gemeinsamen Teilgraphs von Fall und Anfrage benutzt. Beide Entscheidungen können vom Benutzer korrigiert werden.

Für ein generisches Werkzeug zum SFBS sind somit viele wesentlichen Teile vorhanden. Was noch fehlt, ist eine formale Spezifikation der Modellierung der anwendungsabhängigen Erkennungs-, Rekonstruktions- und Vergleichsfunktionen sowie der anwendungsabhängigen Verfeinerung der generischen Heuristiken und Funktionen. Von Expertensystemen ist man gewohnt, daß diese austauschbaren Teile der Wissensbasis deklerativ beschrieben werden. Für das SFBS hieße dies, daß man eine deklerative Sprache vorgeben müßte, in der alle Möglichkeiten, Objekte zu vergleichen, Relationen zu erkennen und zu rekonstruieren und Objekte zur Übertragung auszuwählen, formulierbar wären. Da jedoch sowohl das Format der Objekte als auch die Möglichkeiten, Relationen zu erkennen und zu bearbeiten, nicht einschränkbar sind, würde eine solche Sprache ähnlich komplex wie eine beliebige andere höhere Programmiersprache. Statt einer deklerativen Sprache habe ich deswegen die Signatur der anwendungsabhängigen Funktionen vorgegeben und dem Benutzer freigestellt, welche Programmiersprache er wählt.

Die **Erkennungsfunktionen** R_{typ-b} müssen für ein n-Tupel aus Objekten (O^n) entscheiden, ob eine n-stellige Relation r_{typ-b} gilt. Wenn ja, wird eine Relation r mit dem Typbezeichner typ-b und Verweisen auf die betroffenen Objekte erzeugt. Auf den Typbezeichner wird bei dem Vergleich zugegriffen.

$$ErkennungR_{typ-b} : O^n \rightarrow boolean$$

Von den **Vergleichsfunktionen** wird erwartet, daß sie für zwei gegebene Objekte oder Relationen entscheiden, ob diese kompatibel sind oder nicht.

$$kompatibel : O \times O \rightarrow \{0, 1\}$$

$$kompatibel : r \times r \rightarrow \{0, 1\}$$

Die **Übertragungsheuristik** erhält als Entscheidungsgrundlage alle Information über den Vergleich von Anfrage und Fall. Dies sind die Objektmenge des Falles O^*_{Fall}, die Relationsmenge des Falles R^*_{Fall}, die Objektmenge der Anfrage $O^*_{Anfrage}$, die Relationsmenge der Anfrage $R^*_{Anfrage}$ und die im Matching durchgeführten Zuordnungen $Matching \subset R^*_{Fall} \times R^*_{Anfrage}$. Aus der Zuordnung der Relationen ist ebenfalls die Zuordnung der Objekte ableitbar. Falls Anfrageobjekte zur Korrektur freigegeben wurden, werden auch diese angegeben ($O^*_{Korrektur}$). Das Ergebnis besteht aus den zur

Übertragung ausgewählten Relationen und Objekten des Falles. Die standardmäßige Übertragungsheuristik wurde in Abschnitt 2.3.3 beschrieben.

Auswahl :

$$O^*_{Fall} \times R^*_{Fall} \times O^*_{Anfrage} \times R^*_{Anfrage} \times Matching \times O^*_{Korrektur} \rightarrow O^*_{Fall} \times R^*_{Fall}$$

Die **Rekonstruktionsfunktionen** berechnen die neue Position eines freien Objektes in der Anfrage anhand eines festen Objektes und einer ausgewählten Relation. Das freie Objekt ist entweder ein zu korrigierendes Objekt der Anfrage oder eine Kopie eines aus dem Fall zur Übertragung ausgewählten Objektes. Das Ergebnis ist das veränderte freie Objekt. Die Verwendung dieser Funktionen wurde in Abschnitt 5.2 gezeigt.

$$Rekonstruktion : O^n_{Fest} \times O_{Frei} \times R_{Fall} \rightarrow O$$

Für die vorgestellten binären Relationen gilt n=1. Mehrstellige Relationen könnten mehrere frei Objekte betreffen. Diese müßten nacheinander festgelegt werden, wodurch aufgrund der möglichen verschiedenen Reihenfolgen Backtracking nötig würde. Dieser Aspekt wird jedoch in der Arbeit nicht weiter diskutiert.

7.3 Verfeinerungen des Modells

Um TOPO auf ein Teilgebiet zu spezializieren, kann die Modellierung des anwendungsspezifischen Wissens in allen Bereichen verfeinert werden. Zum einen können zusätzlich zu den definierten Funktionen für die primitiven Relationen aus Kapitel 2 weitere Funktionen definiert werden, die für das Teilgebiet spezifischen Relationen erkennen und rekonstruieren. Zum anderen können die vorgestellten Vergleichsfunktionen für Objekte auf das Teilgebiet angepaßt werden, und schließlich kann Wissen bereitgestellt werden, das entscheidet, welche Relationen eines Falles auf eine Anfrage übertragen werden.

Zusätzliche Erkennungs-/Rekonstruktionsfunktionen können die Qualität des Ergebnisses steigern, falls sie zur Erkennung relevanter Zusammenhänge dienen. Gleichzeitig wird aber der Vergleichsaufwand gesteigert, da die durchschnittliche Zahl der in einer Situation erkannten Relationen, und somit die Komplexität des Strukturgraphens, erhöht wird. Ein Beispiel

ist die für den Innenarchitekten wichtige Relation, daß die Farben zweier benachbarter Objekte in Beziehung stehen. Mithilfe einer solchen Relation kann festgestellt werden, daß die Farben zweier räumlich benachbarter Objekte in einer unüblichen Beziehung stehen. Daraufhin kann die Beziehung fallbasiert durch eine in Fällen häufiger vorkommende Beziehung ersetzt werden.

Für das Klimasystem hingegen ist die Farbe unwesentlich, während die Kapazität einer Leitung eine wesentliche Rolle spielt. Sie betrifft keine benachbarten Objekte, sondern sich berührende (und damit verbundene). Modelliert man eine solche Relation, so können, genauso wie für den Innenarchitekten, unübliche Zusammenhänge erkannt und fallbasiert korrigiert werden.

Um zu verhindern, daß die Farben der Teile des Klimasystems in Beziehung zueinander gesetzt werden, kann man die Relationen objektspezifisch definieren. Dadurch wird zum Beispiel nur die Farbe derjenigen Objekte in Relation zueinander gesetzt, die für den Innenarchitekten wichtig sind. Als eine andere Möglichkeit können je nach Benutzer unterschiedliche Mengen von Relationen benutzt werden. Durch eine derartige Präzisierung der Erkennungsfunktionen würden die zu verarbeitenden Relationsmengen wesentlich kleiner, das Verfahren folglich schneller. Gleichzeitig würde das Ergebnis besser, da es aus präzise definierten Zusammenhängen abgeleitet würde, anstatt aus relativ unspezifischen topologischen Zusammenhängen.

Eine Verfeinerung der Vergleichsfunktion ermöglicht es, in Abhängigkeit des betrachteten Teilgebietes unterschiedliche Objekte und Relationen als kompatibel zu bewerten. Bezüglich der im vorherigen Beispiel genannten Relationen wäre es möglich, auch farblich verschiedene Objekte als kompatibel zu betrachten, da der im Fall vorhandene Zusammenhang zwischen den Farben benachbarter Objekte erkannt und in der Anfrage rekonstruiert werden kann. Das gleiche gilt für die Kapazitäten der Leitungen des Klimasystems. Auch die Vergleichsfunktion für Relationen kann verfeinert werden, um zum Beispiel eine ähnliche Beziehung zu finden, falls es keine identische gibt. In [Freksa, 1992a] wird zum Beispiel eine Distanzfunktion für eindimensionale räumliche Relationen beschrieben. Die Relation „A links von B" hat bei ihm zu der Relation „A überlappt B von links" eine geringere Distanz als zu der Relation „A überdeckt B". Dieser Ansatz wird in Abschnitt 4.2.3 diskutiert.

Eine verfeinerte Vergleichsfunktion bietet nicht nur zusätzliche Möglichkeiten, sondern macht den Vergleich eines Falles mit der Anfrage auch kom-

plexer. Bewerteten die in TOPO benutzten Vergleichsfunktionen nur identische Relationen und Objekte als kompatibel, so muß nach der Verfeinerung abgewägt werden, ob zum Beispiel ein farblich identischer, aber nur ähnlich positionierter Stuhl ähnlicher zu einer Anfrage ist als ein farblich ähnlicher, aber räumlich identisch positionierter Stuhl. Abschnitt 4.2.3 diskutiert dieses Problem.

Eine Spezialisierung der Übertragungsheuristik führt zu präziseren Ergebnissen, so daß weniger unbrauchbare Objekte oder irrelevante Relationen übertragen werden. Man kann im anwendungsspezifischem Wissen angeben, welche Objekte und Relationen übertragen werden in Abhängigkeit der Objekte und Typen, die aus Fall und Anfrage einander zugeordnet wurden. Werden zum Beispiel Stammleitungen und Auslaßöffnungen in Fall und Anfrage einander zugeordnet, so kann das anwendungsspezifische Wissen vorgeben, daß die Verbindungsleitungen aus dem Fall übertragen werden. Für das genannte Komponentenmodell A4, beziehungsweise dessen Nachfolger A5, wird solches Wissen in [Hovestadt, 1995] beschrieben.

Der folgenden Abschnitt wird konkrete Beispiele präsentieren.

7.4 Alternative Anwendungsgebiete

7.4.1 Funktionale Spezifikationen

In [Steinmann, 1994] wird ein Konzept zur funktionalen Spezifikation von Gebäuden vorgestellt. Es ist als Alternative oder Ergänzung herkömmlicher Raumprogramme gedacht (DIN 276/277). Die funktionale Spezifikation eines Gebäudes besteht aus einem Graph (vergleiche Abbildung 7.2). Die Knotenobjekte des Graphen entsprechen Nutzungs- beziehungsweise Struktureinheiten. Die Kanten beschreiben explizit gegebene Relationen zwischen solchen Einheiten. Aufbauend auf diesen Graphen sind einige Werkzeuge zur Erstellung und Verarbeitung funktionaler Spezifikationen entstanden. Sie enthalten einen Vorrat an abstrakten, vorgedachten Planungsobjekten. Dessen beliebige Erweiterung durch einen Benutzer wird von den Werkzeugen ebenfalls unterstützt.

Die nächste Planungsstufe dient der Formfindung. Wie am Beispiel in Abbildung 7.3 gezeigt, soll auf Basis der funktionalen Spezifikation ein geometrisches Modell erstellt werden.

Betrachtet man Abbildung 7.2 als Anfrage und Abbildung 7.3 als Fall, so kann TOPO ohne jegliches anwendungsspezifisches Wissen den Fall mit der

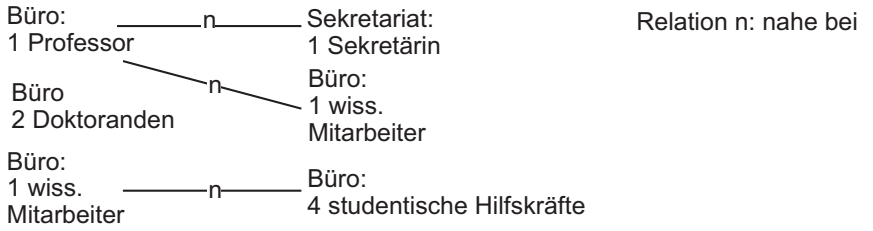

Abbildung 7.2: Diese funktionale Spezifikation zeigt einige Zusammenhänge zwischen den verschiedenen Büros eines zu planenden wissenschaftlichen Instituts.

Anfrage vergleichen, die gemeinsame Teilstruktur finden und Objekte und Relationen aus dem Fall auf die Anfrage übertragen. Dazu gehört auch die Übertragung von Attributwerten, wie zum Beispiel die Raumgröße. Als Vergleichsfunktion reicht der standardmäßige Vergleich auf Identität. Ebenso reicht die generische Übertragungsheuristik aus. Erkennungs- und Rekonstruktionsfunktion werden nicht benötigt, da alle Relationen bereits eingezeichnet sind und eine Anpassung nicht nötig ist. Eine weitere positive Eigenschaft ist, daß hinzukommende Objekt- oder Relationstypen TOPO keinerlei Schwierigkeiten bereiten, da TOPO sie nur auf Identität prüft. Im Prinzip entspricht eine derartige Benutzung von TOPO einer Datenbankanfrage, die ja auch ohne anwendungsspezifisches Wissen auskommt. Nur ist in einer Datenbank ein Graphvergleich nicht möglich.

Verzichtet man auf eine automatische Übertragung, so liefert die Benutzung von TOPO noch weit mehr Information als bisher in diesem Abschnitt gezeigt. Findet man mit TOPO zum Beispiel eine ähnliche funktionale Spezifikation, so kann man sich das zum Fall gehörende geometrische Modell ansehen. Falls das im Fall beschriebene Gebäude schon gebaut wurde, bietet sich eine Vielzahl an Möglichkeiten:

- Man kann sich das Gebäude ansehen und überlegen, ob einem die Ausführung gefällt, was einem nicht gefällt und was man folglich anders spezifizieren sollte.
- Man kann die Folgekosten, also den Unterhalt abschätzen.
- Falls die tatsächlichen Baukosten weit über dem liegen, was man bereit ist zu investieren, kann man Rückschlüsse ziehen, wie man die Spezifikation ändern könnte.

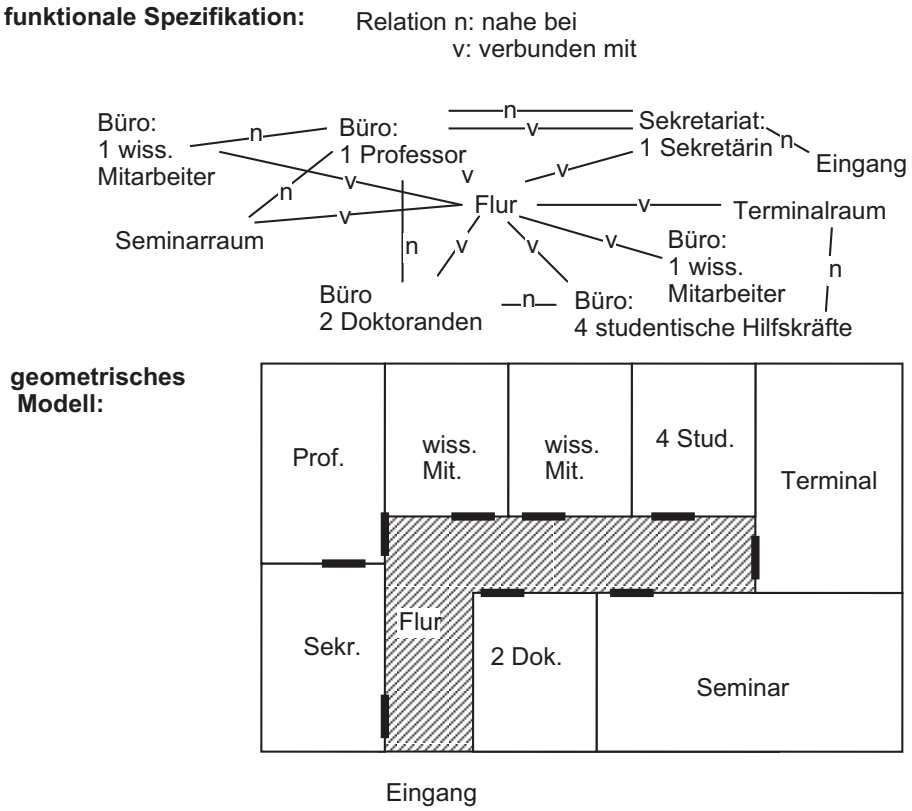

Abbildung 7.3: Für eine funktionale Spezifikation wird im nachfolgenden Planungsschritt ein geometrisches Modell erstellt.

Um TOPO noch besser auszunutzen, kann anwendungsspezifisches Wissen definiert werden. Dieses könnte zum Beispiel beim Vergleich

- Ähnlichkeiten zwischen Objekten erkennen und benutzen
- oder Zusammenhänge zwischen Seminarraumgröße und Anzahl der Institutsmitarbeiter erkennen und in einer Anfrage rekonstruieren
- oder die Kosten an die Anzahl und Art der Räume anpassen

Die Vielzahl der Möglichkeiten ist nur durch die Bereitschaft begrenzt, das Wissen zu modellieren.

7.4.2 Modellierung nicht-orthogonaler Relationen im dreidimensionalen Raum

Abbildung 7.4 zeigt ein Anwendungsbeispiel aus dem Bereich der geographischen Informationssysteme. An ihm werde ich ein Modell für nicht-orthogonale Relationen vorstellen, welches jedoch im Rahmen dieser Arbeit nicht implementiert wurde.

Die Anfrage soll mit den Bebauungsplänen a, b und c verglichen werden, um eine passende Parzelleneinteilung für das mit einem Fragezeichen markierte Gebiet zu erhalten. In den drei Fällen sind Teile der Parzelleneinteilung schwarz hervorgehoben. In diesem Beispiel wird eine nicht-orthogonale Modellierung nur für die horizontalen Dimensionen benötigt, um mit Drehungen in der horizontalen Ebene umzugehen. Die dritte Dimension einer Relation wird als orthogonal dazu angenommen, wie es für die Architektur oder Landschafts-/Straßenplanung sinnvoll ist.

Die Modellierung räumlicher Relationen im nicht-orthogonalen Layout ist wesentlich komplizierter als im orthogonalen Fall. Dafür gibt es zwei Ursachen. Zum einen ist die Form der Objekte selber nicht mehr ausschließlich rechteckig, wodurch der die Topologie beschreibende Teil der Relationen wesentlich komplizierter werden kann. Zum anderen gibt es keine standardmäßige Ausrichtung der Objekte, wodurch der Vergleich der Richtung zweier Relationen ebenfalls schwieriger ist als im orthogonalen Fall. Den dritten Aspekt, daß Beziehungen zwischen Formen von Objekten wichtig sein können, werde ich nicht diskutieren, da dies nur in Ausnahmefällen eine Rolle spielt.

Im Vergleich zu rechtwinkligen, an den Koordinatenachsen ausgerichteten Objekten gibt es zum Beispiel mehrere Arten, auf die sich zwei Objekte überlappen können. Abbildung 7.5 zeigt einige Beispiele. Von links nach

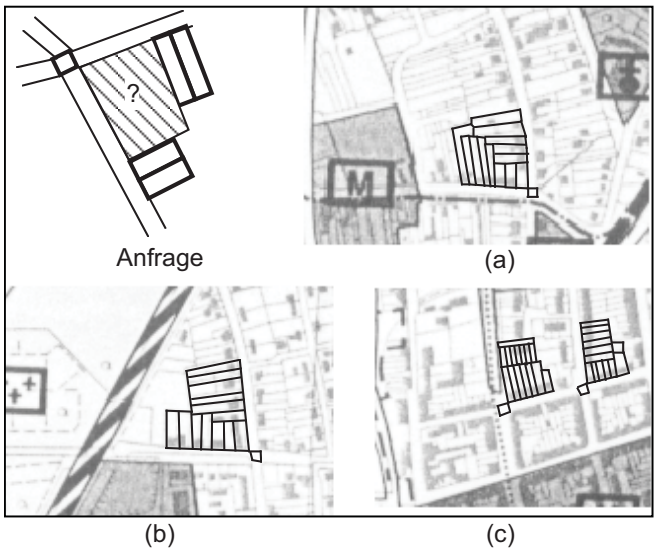

Abbildung 7.4: Ein Beispiel aus dem Anwendungsbereich von GIS.

rechts nimmt der Grad der Überlappung zu. Ganz links überlappen sich die umschließenden Rechtecke in einem vorgegebenen orthogonalen Raster. Daneben überlappen sich die minimalen konvexen Hüllen. Noch weiter rechts überlappt sich schon ein Teil der Innenräume, und ganz rechts wird der Schwerpunkt des linken Objektes vom Innenraum des rechten überlappt. Das umschließende Rechteck der Ellipse ist dort schon ganz in dem umschließenden Rechteck des anderen Objektes enthalten. In [Papadias *et al.*, 1995] werden allein schon für die schwächste Form der Überlappung mehrere verschiedene Möglichkeiten im zweidimensionalen Raum gezeigt. In einer konkreten Anwendung müßten diese gegebenenfalls unterschieden werden. Zum Beispiel würden die rechten beiden Überlappungen als verbotene Kollision interpretiert und die linken als räumliche Nähe.

Für das zweite Problem gibt es keine so eindeutige Lösung. Um zwei Relationen zu vergleichen, muß die Richtung einer Relation eindeutig bestimmbar sein. Außerdem sollte sich bei geringfügigen Transformationen eines oder beider Objekte die Richtung ebenfalls nur geringfügig verändern. Um auch Spiegelungen und Verdrehungen von Teilsituationen zu erkennen, müssen deren Auswirkungen auf die Richtung einer Relation eindeutig und

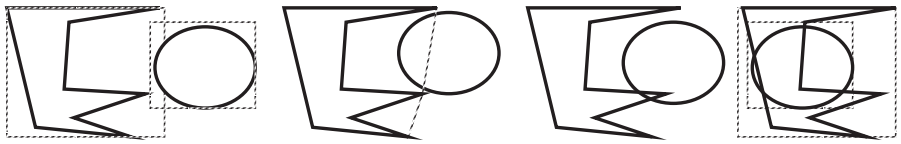

Abbildung 7.5: Objekte können sich auf viele verschieden Arten überlappen.

damit erkennbar sein. Ich habe zwei verschiedene Ansätze in der Literatur gefunden, die nicht-orthogonale Richtungen von Relationen untersuchen.

[Freksa, 1992b] und [Latecki, 1993] gehen davon aus, daß jedes Objekt eine eindeutige Ausrichtung hat, zum Beispiel die Front eines Hauses oder Autos. Relativ zu diesen lokalen Ausrichtungen beschreiben sie die Richtung einer Relation, zum Beispiel: „Rechts vom Auto steht ein Gebäude". In GIS-Anwendungen gibt es jedoch nur selten eindeutige objektspezifische Ausrichtungen, weswegen dieser Ansatz nicht verwendbar ist.

Auch wenn die Objekte nicht mehr an den Koordinatenachsen ausgerichtet sind, kann man sie dennoch als Bezugspunkte benutzten. [Hernandez, 1992] beschreibt die Richtung einer Relation relativ zu einem globalen Koordinatensystem. Bei ihm gibt es acht verschiedene Richtungen, die jeweils 45° abdecken: vor, rechts vor, rechts, rechts hinter, hinter, links hinter, links und links vor. Dieser Ansatz bestimmt die Richtung einer Relation eindeutig, und es lassen sich auch Spiegelungen und Drehungen um Vielfache von 45° erkennen. Ein geringfügige Verschiebung eines Objektes kann jedoch dazu führen, daß sich die Richtung stark verändert, weil sie dicht an der Grenze zweier 45°-Segmente liegt. Zum anderen führen Drehungen, deren Drehungswinkel kein Vielfaches von 45° sind, zu unterschiedlichen Veränderungen von Relationen. Liegt etwa eine Relation weniger als 5° von einer Segmentgrenze entfernt, so führt eine Drehung um 5° zu einer Veränderung der Relation. Liegt sie hingegen mehr als 5° entfernt, so verändert sich die Richtung nicht. In meinem Vorschlag werde ich die diskrete Einteilung in Richtungssegmente durch eine kontinuierliche ersetzen und zeigen, wie damit das GIS-Anwendungsbeispiel bearbeitet werden kann.

Wie Abbildung 7.6 zeigt, definiere ich die Richtung einer Relation durch einen Strahl vom Schwerpunkt des einen zum Schwerpunkt des anderen Objektes. Genauso wie bei der Modellierung bei TOPO wird beachtet, ob die Richtung von A nach B geht oder umgekehrt, wobei wieder gilt: $ARB = BR^{-1}A$ (also zum Beispiel: A vor B = B hinter A, vergleiche Ab-

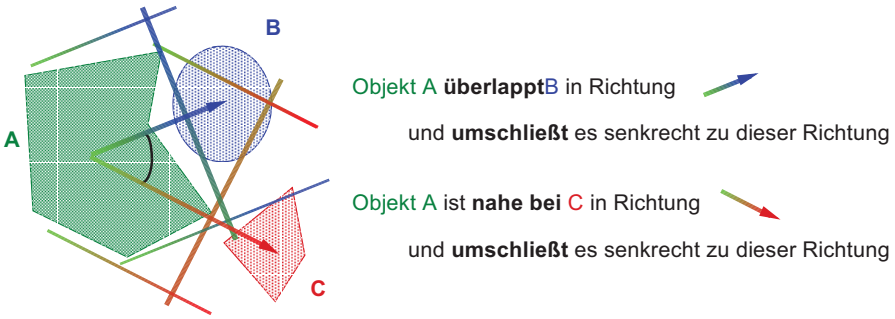

Abbildung 7.6: Die Richtung einer Relation wird durch einen Strahl vom Schwerpunkt des einen zum Schwerpunkt des anderen Objektes bestimmt und die Topologie durch die Beziehung der zu dieser Richtung orthogonalen umschließenden Rechtecke.

schnitt 2.3.1). Um die Topologie zu beschreiben, werden in Richtung des Strahls umschließende Rechtecke für beide Objekte berechnet und deren Topologie parallel und orthogonal zum Strahl mit den schon für TOPO verwendeten acht Relationen beschrieben. Eine dritte räumliche Relation beschreibt die vertikale Beziehung. Damit besteht eine Relation aus einem Strahl und drei topologischen Relationen.

Um zwei räumliche Situationen zu vergleichen, muß noch spezifiziert werden, wann zwei Relationen kompatibel sind. Zum einen müssen die drei topologischen Relationen identisch sein. Es macht jedoch keinen Sinn zu fordern, daß auch die Richtung identisch sein muß. Im Gegenteil sollen ja auch verdrehte Situationen wiedererkannt werden (vergleiche Abbildung 7.8). Ähnlich wie bei TOPO relative Positionen statt absoluter Positionen verglichen werden, müssen hier relative Richtungen statt absoluter Richtungen verglichen werden. Das erste einander zugeordnete Paar von Relationen bestimmt die Matching-Richtung. Das zweite zugeordnete Paar legt fest, ob der zugeordnete Teil der Anfrage eine Spiegelung eines Teiles des Falles ist. Abbildung 7.7 verdeutlicht den Effekt. Bei der Anfrage werden alle Winkel im Uhrzeigersinn gemessen. Entsprechend dem zweiten zugeordneten Paar von Relationen wird im Fall im folgenden genauso, oder bei einer Spiegelung, gegen den Uhrzeigersinn gemessen.

Nach der Festlegung der Match-Richtung und der Richtung der Winkelmessung kann die relative Richtung zweier Relationen anhand ihrer Strahlen

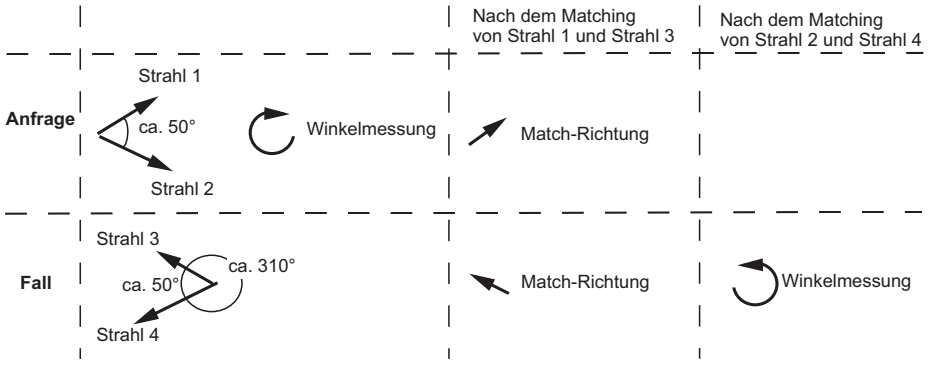

Abbildung 7.7: Das erste zugeordnete Paar von Relationen bestimmt die Matching-Richtung. Das zweite zugeordnete Paar legt fest, ob der Winkel im Fall gegen oder mit dem Uhrzeigersinn gemessen wird.

verglichen werden. Wie exakt die Richtungen übereinstimmen sollten, damit sie als kompatibel bewertet werden, hängt von der Anwendung ab. Man müßte es in der Praxis ausprobieren. Als ersten Versuch würde ich eine Differenz von bis zu der Hälfte eines 45°-Winkels zulassen, also 22,5°, was einer Auflösung von 16 verschiedenen Richtungen im zweidimensionalen Raum entspricht.

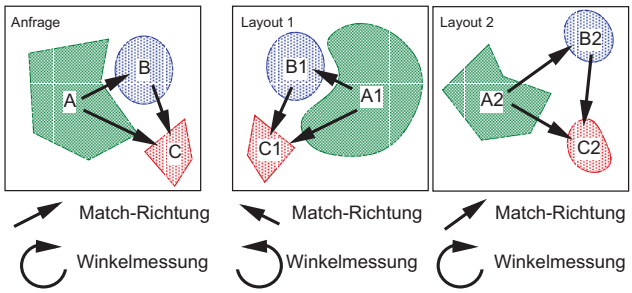

Abbildung 7.8: Ein Beispiel für den Vergleich von nicht-orthogonalen Layouts.

Als Beispiel sollen die beiden Layouts 1 und 2 (Abbildung 7.8) mit der Anfrage verglichen werden. Bei der ersten Zuordnung einer Relation aus

dem Fall zu einer aus der Anfrage spielt die Richtung noch keine Rolle. Sei die Relation $\overline{AB}$ zwischen A und B der Relation $\overline{A1B1}$ zwischen A1 und B1 zugeordnet worden. Ordnet man als nächstes die Relationen $\overline{AC}$ und $\overline{A1C1}$ einander zu, so liegt damit fest, daß in Layout 1 die Winkel gegen den Uhrzeigersinn gemessen werden. Im Anschluß kann die dritte Relation der Anfrage $\overline{BC}$ einer Relation aus Layout 1 ($\overline{B1C1}$) zugeordnet werden. Auch im zweiten Layout lassen sich alle Relationen den Relationen der Anfrage so zuordnen, daß die Richtungen kompatibel sind. Allerdings sind die Relationen $\overline{AB}$ und $\overline{A2B2}$ nicht kompatibel, da die umschließenden Rechtecke der Objekte A und B sich im Gegensatz zu den Objekten A2 und B2 überlappen. Deshalb ist Layout 1 ähnlicher zur Anfrage als Layout 2.

Nach diesen abstrakten Beispielen nun zurück zur GIS-Anwendung. Abbildung 7.9 zeigt die Parzellen und Kreuzungen aus Abbildung 7.4. Zusätzlich sind die Strahlen der Relationen eingezeichnet, die Bestandteil des jeweils größten Matchings sind.

Nach der Zuordnung der Relationen aus Fällen und Anfrage soll die Position der zwischen Kreuzung und zugeordneten Parzellen gelegenen Parzellen auf die Anfrage übertragen werden. Abbildung 7.10 zeigt dies exemplarisch für die Fälle (a) und (b). Zur Veranschaulichung zeigt die erste Spalte eine Überlagerung der Objekte und Strahlen von Fall (grau) und Anfrage (schwarz). Zu dem Zweck wurden die Fälle entsprechend den unterschiedlichen Matching-Richtungen und Längen der Strahlen gedreht und skaliert. Dieser Schritt dient nur zur Veranschaulichung. Er wird nicht zur Anpassung benötigt, da die Richtungen und Längen der Strahlen ja nicht absolut, sondern relativ zur Matching-Richtung und Objektgröße repräsentiert werden. In der zweiten Spalte sind die Strahlen zur besseren Übersicht ausgeblendet. Die dritte Spalte zeigt das Ergebnis der Rekonstruktion der in den Fällen vorhandenen Relationen und Objekte in der Anfrage. Genauso wie im orthogonalen Layout wird je eindimensionaler Relationen eine vom Typ der Relation abhängige Rekonstruktionsfunktion aufgerufen. Bei der Rekonstruktion der schraffierten Parzelle aus Fall (a) (Abbildung 7.10) mußte die Größe stark verändert werden, um die Überlappungs-Relation zu den unteren Parzellen und die Enthält-Relation zu den rechten Parzellen zu erhalten. Die starke Veränderung könnte darauf hindeuten, daß das Ergebnis einige positive Eigenschaften des Falles nicht mehr enthält. Dies trifft tatsächlich zu, denn die Größen der Parzellen waren ehemals ungefähr gleich und ergaben sinnvolle Grundstücke, während im neuen Ergebnis zu kleine und zu große Grundstücke entstanden sind.

Die Rekonstruktion der Parzellen aus Fall (b) hat die Parzellengrösse

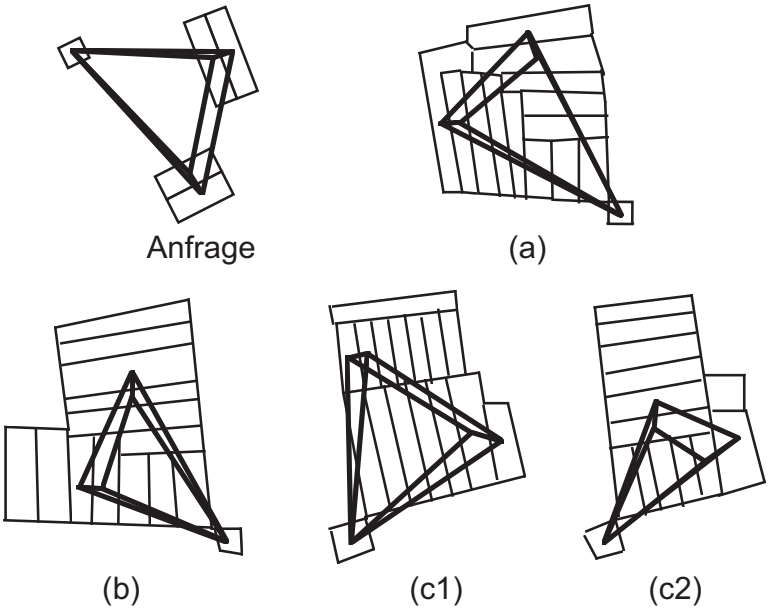

Abbildung 7.9: Dies Bild zeigt die Anfrage, bestehend aus einer Kreuzung und 4 Wohnparzellen, und die Objekte der Fälle. Die dicken Striche repräsentieren die Strahlen, die einander zugeordnet werden können.

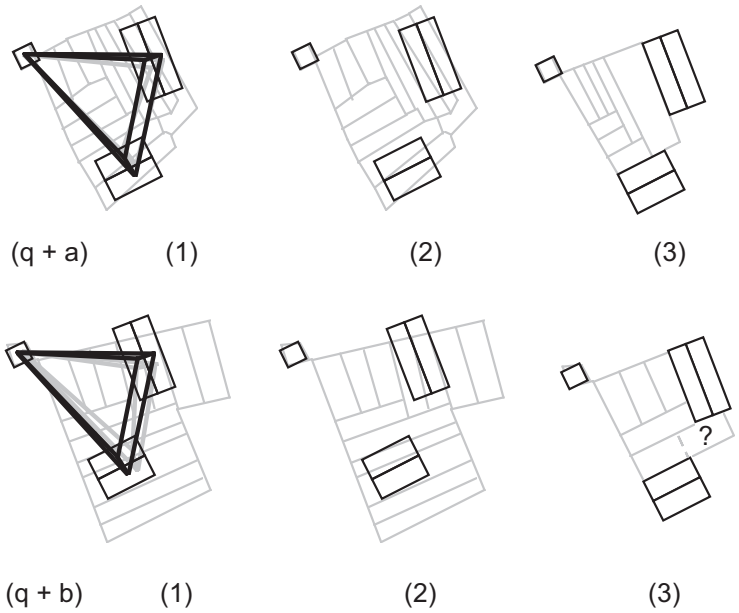

Abbildung 7.10: In diesem Bild sind die Ausschnitte (a) und (b) in die Anfrage integriert. Dazu wurden die Graphen rotiert und skaliert (1). Nach Ausblendung der Graphen (2) kann man die Situation besser erkennen. (3) zeigt das Ergebnis der Anpassung.

weniger verändert. Für den mit dem Fragezeichen gekennzeichneten Bereich ist jedoch nicht eindeutig feststellbar, ob er zu der links davon liegenden Parzelle gehört oder freibleibt. Entsprechend der Gleich-Relation zur unteren Parzelle dürfte der ?-Bereich nicht zur rekonstruierten Parzelle gehören. Die Überlappungs-Relation zu den rechten Parzellen würde es jedoch erfordern. Um diese Frage zu klären, benötigt man zusätzliches Wissen. Ohne solches Wissen könnten einfach beide Alternativen dem Benutzer angeboten werden.

Zusammenfassung der Antworten:

1. $\Rightarrow$ Das SFBS eignet sich prinzipiell für alle Anwendungsgebiete, für die eine Fallbasis vorhanden oder erzeugbar ist und natürlicherweise Struktur, also Abhängigkeiten zwischen Objekten, eine Rolle spielt.
2. $\Rightarrow$ Je dichter eine Fallbasis den Raum der möglichen Anfragen abdeckt, desto weniger Anpassung ist erforderlich und desto besser wird das Ergebnis sein. Die Fälle sollten möglichst Ausgangssituationen und Ergebnisse einzelner Arbeitsschritte beinhalten, wodurch automatisch nur sinnvolle Übertragungen durchgeführt würden.
3. $\Rightarrow$ Alle Bestandteile von TOPO, bis auf die Erkennungs- und Rekonstruktionsfunktionen, sind generisch, und selbst diese können zur Verarbeitung beliebiger orthogonaler Layouts verwendet werden.
4. $\Rightarrow$ Alle vier Bestandteile des anwendungsspezifischen Wissen von TOPO können verfeinert werden.
5. $\Rightarrow$ In diesem Kapitel wurden die Verarbeitung funktionaler Spezifikationen und GIS-Pläne als Anwendungsbereiche vorgestellt.

Kapitel 8

Vergleich zu anderen Ansätzen

Fragen:

1. Welche Eigenschaften besitzen andere Ansätze zum Vergleich von Strukturen?

2. Gibt es andere Ansätze zum fallbasierten Schließen, die auch Struktur repräsentieren, und wie lassen diese sich mit dem SFBS vergleichen?

In diesem Kapitel werden fremde Ansätze und Konzepte beschrieben, die sich im Kern auch mit dem Vergleich von Strukturen befassen. Der erste Abschnitt gibt eine Abgrenzung des SFBS zum Konzept des analogen Schließens. Abschnitt 8.2 wird einige Ansätze unter dem Aspekt einer interessanten Realisierung des Strukturvergleiches beschreiben und jeweils mit dem in dieser Arbeit vorgestellten Konzept vergleichen. Abschnitt 8.3 zeigt eine Verwendung des von mir benutzten Graphmatchingverfahrens in einer völlig anderen Anwendung. Der letzte Abschnitt 8.4 skizziert einige von den Konferenzen der letzten Jahre her bekannte Ansätze, wobei offensichtlich wird, daß sie alle in das von mir vorgestellte Konzept des SFBS passen. Im Gegensatz zum SFBS sind sie jedoch auf einzelne Anwendungsbereiche beschränkt, beziehungsweise benötigen viel anwendungsspezifisches Wissen. Außerdem wird dieses Wissen nicht generisch spezifiziert und die Funktionalität der Ansätze deckt meistens nur einen Bruchteil der Funktionalität des SFBS ab.

8.1 Vergleich zum analogen Schließen

Wie im fallbasierten Schließen wird auch im analogen Schließen [Hall, 1989] Erfahrungswissen zur Problemlösung benutzt. Der Begriff des analogen Schließens ist wesentlich älter als der des fallbasierten Schließen, erste Überlegungen wurden schon 1945 veröffentlicht [Polya, 1945]. Die ersten Ansätze zum fallbasierten Schließen [Schank, 1982] wurden entsprechend als Spezialform des analogen Schließen angesehen. Bis heute gibt es keine allgemein anerkannte und eindeutige Unterscheidung der Begriffe. Es gibt eher eine Reihe von Eigenschaften, die man eher dem einen oder dem anderen Gebiet zuordnet:

- Das analoge Schließen beschäftigt sich typischerweise mit gebietsübergreifenden Vergleichen [Aamodt und Plaza, 1994], zum Beispiel daß der Strom- und der Wasserkreislauf vergleichbar sind.

- Im analogen Schließen werden Konzepte übertragen. Dazu wird ein neues Problem so weit abstrahiert, bis es mit der entsprechend abstrakten Darstellung eines Falles identisch ist. Ist zum Beispiel bekannt, daß man zur Erhöhung des Wasserdurchflusses in einem Rohr den Druck steigern muß oder das Rohr verbreitern kann, so könnte analog geschlossen werden, daß man zur Erhöhung eines Stromflusses die Spannung erhöhen oder das Kabel verbreitern kann. Dieser hohe Abstraktionsgrad ist laut [Gentner, 1983] typisch für die Art menschlichen Problemlösens, die durch analoges Schließen nachemfunden werden soll.

- Ein zentraler Bestandteil des analogen Schließens ist die dynamische Zuordnung der Elemente einer Anfrage zu denen eines Falles, auch „Mapping" genannt.

- Schwerpunkt fallbasierter Ansätze ist häufig die effiziente Verarbeitung von Fällen. Dabei bildet komplexitätsbedingt die explizite Repräsentation und Verarbeitung von Strukturen eher eine Ausnahme. Strukturelle Eigenschaften werden abstrahiert und durch Attribute repräsentiert.

- Im fallbasierten Schließen werden umfangreiche Fallbasen als positiv betrachtet, um einen möglichst gut passenden Fall zu finden, dessen Anpassung wenig Aufwand erfordert. Gebietsübergreifende Verwendung von Fällen kommt nicht vor. Fälle sind häufig Zwischenergeb-

nisse üblicher Arbeitsschritte und werden zur Unterstützung derselben Arbeitsschritte verwendet.

Genau wie das analoge Schließen basiert das SFBS auf einer Zuordnung von Elementen aus Anfrage und Fall. Der zur Zuordnung benutzte Abstraktionsgrad ist beim SFBS jedoch fest vorgegeben und eher niedrig. Dies wird zum Beispiel an der Einfachheit der benutzten Relationen deutlich. Da außerdem die Anfrage und die Fälle aus einem Anwendungsgebiet stammen, Zwischenergebnisse der unterstützten Arbeitsschritte repräsentieren und der Schwerpunkt des SFBS eher effizienzorientiert ist, bezeichne ich es als ein Konzept zum fallbasierten Schließen.

8.2 Interessante Strukturvergleiche

Da der Vergleich zweier Graphen np-vollständig ist, sollten möglichst wenige Fälle der Fallbasis mit der Anfrage auf diese Weise zu verglichen werden. Zu diesem Zweck enthalten einige Ansätze eine auf Strukturvergleiche abgestimmte Fallbasis. Beispiel hierfür sind die in 8.2.1 und 8.2.2 beschriebenen Ansätze.

Auch der in Abschnitt 8.2.3 beschriebene Ansatz SYN enthält eine effiziente Organisation der Fallbasis. Sein Schwerpunkt ist jedoch die Ausnutzung der Vorteile, die die Einschränkung der betrachteten Graphen auf Bäume bietet. Ein weiterer besonderer Aspekt von SYN ist die gleichzeitige Benutzung aller Fälle einer ganzen Fallklasse zur Anpassung.

8.2.1 MACS: Clusterbildung

Der im FABEL-Projekt von [Tammer *et al.*, 1995] entwickelte Ansatz schlägt vor, die Fallbasis zu clustern. Strukturell verschiedene Fälle, deren strukturelle Ähnlichkeit einen Schwellwert übersteigt, werden zu einem Cluster zusammengefaßt und durch ihre größte gemeinsame Teilstruktur repräsentiert. Strukturelle Ähnlichkeit wird dabei durch die Größe der größten gemeinsamen Teilstruktur bestimmt.

Durch diesen Ansatz müssen nicht mehr alle Fälle mit einer Anfrage verglichen werden, sondern nur noch die Repräsentanten. Der Nachteil ist, daß der Ansatz nicht garantiert, daß der Fall mit der größten strukturellen Ähnlichkeit zur Anfrage gefunden wird. Außerdem kann nicht garantiert werden, daß die Strukturierung der Fallbasis in zumutbarer Zeit abgeschlossen ist, da je zwei Strukturen miteinander verglichen werden müssen.

Im Gegensatz zu diesem Ansatz wird der np-vollständige Vergleich beim SFBS durch approximative Vergleiche (vgl. 4.3) vermieden oder zumindest zeitlich begrenzt und die Organisation der Fallbasis dem Retrieval-Werkzeug ASPECT überlassen. Dabei garantiert ASPECT im Gegensatz zu MACS, daß es keine falschen Aussagen über die Rangfolge der Fälle bezüglich ihrer Ähnlichkeit zur Anfrage macht.

8.2.2 Speicherkapazität statt Rechenzeit

Bunke und Messmer [Bunke und Messmer, 1994] organisieren die Fälle in einem Entscheidungsbaum, der sämtliche Teilgraphen aller Fälle der Fallbasis enthält. Die Wurzel ist der leere Graph. Alle Graphen der n-ten Ebene beinhalten n Knoten und sind verbunden mit allen Teilgraphen der höheren Ebenen, die sie enthalten (vergleiche Abbildung 8.1). Für einen Anfragegraphen werden von der Wurzel aus alle Wege entlang der in der Anfrage enthaltenen Teilgraphen verfolgt. Angekommen auf Ebene n ist die Fallbasis auf diejenigen Fälle eingeschränkt, die alle Teilgraphen der Größe n der Anfrage enthalten. Die Suchzeit ist somit unabhängig von der Anzahl der Fälle und hängt nur von der Anzahl der enthaltenen verschiedenen Teilgraphen ab. Deswegen eignet er sich besonders für große Fallbasen mit sehr ähnlichen Fällen.

Das Hauptproblem ist der im schlimmsten Fall exponentielle Speicherbedarf. Der Ansatz ist vergleichbar mit der Verwendung von komplexen Relationen (Abschnitt 4.2.2), da ebenfalls größere, in mehreren Fällen gemeinsam vorkommende Teilgraphen explizit repräsentiert werden. Dort wird das Speicherplatzproblem jedoch dadurch gemildert, daß nur markante Teilgraphen (Gestalten) gespeichert werden.

8.2.3 SYN: Hierarchische Fallbasisorganisation, Vergleich eingeschränkter Graphen und gleichzeitige Anpassung mehrere Fälle

SYN [Börner *et al.*, 1996] ist ein auf Baumstrukturen spezialisiertes Werkzeug, welches seine Fallbasis hierarchisch organisiert und zur Anpassung eine ganze Fallklasse statt eines einzelnen Falles verwendet.

Ein Anwendungsbeispiel sind Zuluftsysteme, deren Baumstruktur durch die eindeutige Identifizierung der Stammleitung als Wurzel gegeben ist. Die Einschränkung auf Baumstrukturen bietet erhebliche Vorteile gegenüber der Verarbeitung uneingeschränkter Graphen:

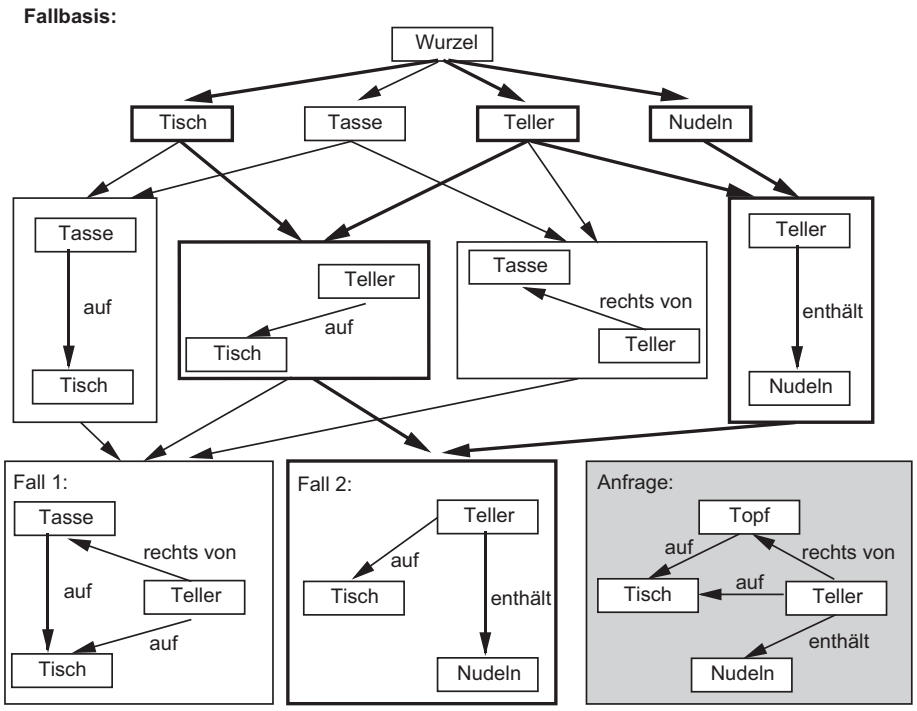

Abbildung 8.1: [Bunke und Messmer, 1994] Die untere Zeile der Abbildung zeigt zwei Fälle und eine Anfrage. Die Fallbasis besteht aus allen Teilgraphen der Fälle. Die fett gezeichneten Pfeile und Rechtecke beschreiben die Pfade, die bei der Suche nach dem zur Anfrage größten Teilgraphen verfolgt werden.

- Der größte gemeinsame Teilbaum zweier Bäume ist eindeutig unter der Annahme, daß die Wurzel enthalten sein muß. Zwar muß die größte gemeinsamer Teilstruktur aus Sicht des Benutzer nicht die beste sein, aber die Eindeutigkeit liefert eine begründbare Heuristik zur Auswahl des Matchings.

- Aufgrund der eindeutigen Wurzel kann die Fallbasis, beginnend mit der Wurzel, hierarchisch in einem Entscheidungsbaum organisiert werden, dessen Speicherplatzbedarf nur quadratisch von der Anzahl der Fälle und linear von der Tiefe der enthaltenen Bäume abhängt. Im Gegensatz dazu müssen bei allgemeinen Graphen alle Teilgraphen der Fälle der Fallbasis im Entscheidungsbaum repräsentiert werden, was abhängig von der Größe der enthaltenen Graphen zu einem exponentiellem Speicherplatzbedarf führt (vgl. 8.2.2).

- Aufgrund des Wissens über die Baumstruktur kann SYN entscheiden, wann ein Baum vollständig ist. Für eine Anfrage mit Zuluftelementen müssen ausschließlich diejenigen Elemente aus einem Fall auf die Anfrage übertragen werden, die die Anfrageobjekte mit der Wurzel (Stammleitung) verbinden. Ein ähnlich generisches Kriterium zur Bewertung der Vollständigkeit einer Lösung ist für allgemeine Graphen nicht formulierbar.

Wie gesagt, organisiert SYN die Fallbasis hierarchisch. Die Knoten entsprechen den Repräsentanten der Fallklassen (vergleiche Abschnitt 8.2.1). Die Blätter des Fallbasisbaumes bilden die Fälle. Sukzessive bekommen die zwei ähnlichsten Knoten des Baumes, die noch keinen Vaterknoten besitzen, einen neuen Vaterknoten. Der oberste Vaterknoten bildet die Wurzel des Baumes. Die mit einem Vaterknoten verbundenen Blätter bilden dessen Fallklasse.

Für eine Anfrage wird in der Fallbasis, beginnend mit der Wurzel, nach dem Knoten mit der größten zur Anfrage identischen Teilstruktur gesucht. Die in den Fällen einer Fallklasse enthaltenen Objekte, die nicht Bestandteil der zugeordneten Knoten sind, bilden die möglichen Erweiterungen eines Repräsentanten. Jede einzelne Erweiterung besitzt eine Wahrscheinlichkeit, die ihrer Häufigkeit in der Fallklasse entspricht. SYN überträgt diejenigen Erweiterungen, die eine möglichst kleine, aber vollständige Struktur als Lösung ergeben.

Für Baumstrukturen ist der in SYN benutzte Strukturvergleich dem allgemeinen Graphvergleich von TOPO überlegen und sollte in passenden An-

wendungsbereichen den Graphvergleich in TOPO ersetzen. Der größte Nachteil von SYN ist jedoch gerade seine eingeschränkte Anwendbarkeit.

8.3 Bestimmung gemeinsamer Teiltopologien in Proteinstrukturen

[Koch *et al.*, 1996] benutzt den Algorithmus von [Bron und Kerbosch, 1973] zur Bestimmung der größten gemeinsamen Teiltopologien in Proteinstrukturen. Neben der interessanten Anwendung beschreibt [Koch *et al.*, 1996] auch eine Evaluation verschiedener Varianten des Bron-Kerbosch-Algorithmus. In ihrer Anwendung werden durch Sekundärstrukturelemente repräsentierte Proteinstrukturen verglichen. Diese Sekundärstrukturen abstrahieren von der Komplexität der atomaren Beschreibung der Proteine (vergleiche Abbildungen 8.2 und 8.3). Sie sind jedoch ausreichend, um die wesentliche Faltungsmuster eines Proteins zu charakterisieren.

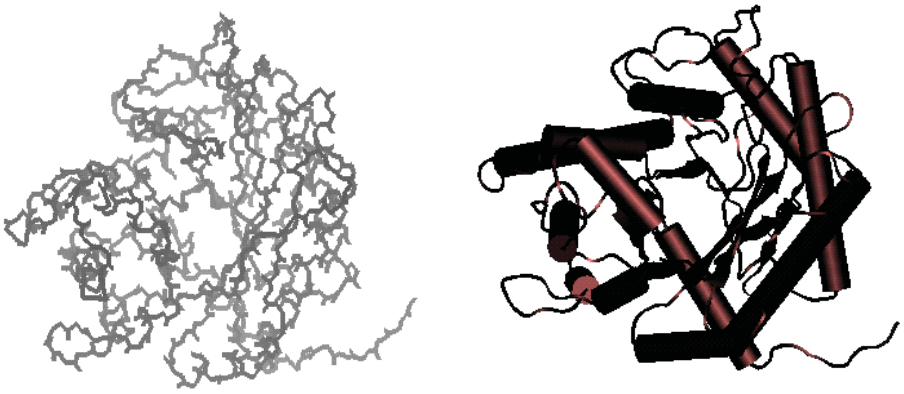

Abbildung 8.2: Links sieht man die atomare Struktur der Rückgradatome eines Aldolase-Moleküles. Rechts daneben dessen Sekundärstruktur in Zylinder-Pfeil-Darstellung.

Im Gegensatz zur beschriebenen Anwendung im Bauwesen gibt es kaum unterschiedliche Relationstypen. In [Koch *et al.*, 1996] werden drei Relationstypen zwischen den Sekundärstrukturelementen unterschieden (siehe Abbildung 8.3). Zum einen bilden die Verbindungen der Sekundärstrukturele-

mente des Moleküls in der schematischen Darstellung sequentielle Kanten. Die anderen beiden Kantentypen verbinden parallele und antiparallele Sekundärstrukturelemente.

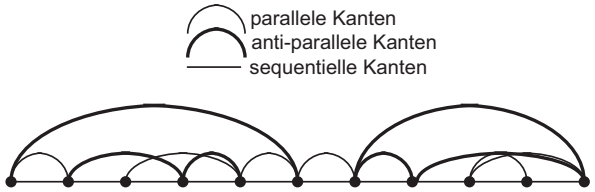

Abbildung 8.3: Dieses Bild zeigt eine schematische Darstellung des Aldolase-Moleküles aus Abbildung 8.2.

Der Vergleich von Proteinstrukturen ist eine grundlegende Aufgabe in der Analyse von Proteinen. Die Aussagekraft der Identität einiger Teilstrukturen zweier Proteine ist hingegen noch nicht vollständig anerkannt. Gemeinsame Teilstrukturen können dem Experten jedoch Hinweise auf evolutionäre Verwandtschaft oder auf ähnliche Funktionalität geben.

Dieser Ansatz zum Vergleich von Proteinen wurde in diesem Kapitel genannt, da er ein Beispiel für eine ganz andere Anwendung zeigt, die ebenfalls sehr komplex ist, wo die Struktur eine große Rolle spielt, und wo TOPO direkt einsetzbar wäre. Leider ist die Formalisierung und Bedeutung von Struktur in diesem Anwendungsgebiet noch so schwach erforscht, daß eine brauchbare automatische Anpassung noch nicht realisierbar ist.

8.4 Einordnung strukturorientierter Ansätze

Wie die vorherigen Abschnitte gezeigt haben, ignorieren bisherige Ansätze keineswegs die Struktur. Sie sind jedoch alle nur sehr eingeschränkt zu benutzten und wagen es nicht, allgemeingültige Anforderung an das Wissen und seine Funktionalität zu formulieren. Von der Art sind auch die folgenden Ansätze, die ich aufgrund ihres Bekanntheitsgrades vorstelle und mit dem SFBS vergleiche.

8.4.1 MoCAS/PARIS: hierarchische Anpassung

In [Bergmann *et al.*, 1994] wird ein Ansatz zur regelbasierten Anpassung vorgestellt, welcher in dem Diagnosesystem MoCAS und dem Planungssystem PARIS realisiert wurde. Beide Systeme bekommen als Anfrage eine Menge von Fakten und erzeugen eine Struktur aus Fakten und Regeln. Bei der Diagnose entspricht diese Struktur einer Diagnose inklusive Herleitung von Symptomen und bei der Planung einem Plan aus Aktionen und Zuständen.

Sowohl für Fakten als auch für Regeln existiert ein Modell, welches Abstraktion erlaubt. Für die Fakten bedeutet dies, daß mehrere Fakten zusammengefaßt werden können und für die Regeln, daß eine Teilstruktur aus Regeln und Fakten zu einer komplexen Regel zusammengefaßt werden kann (vergleiche Abbildung 8.4).

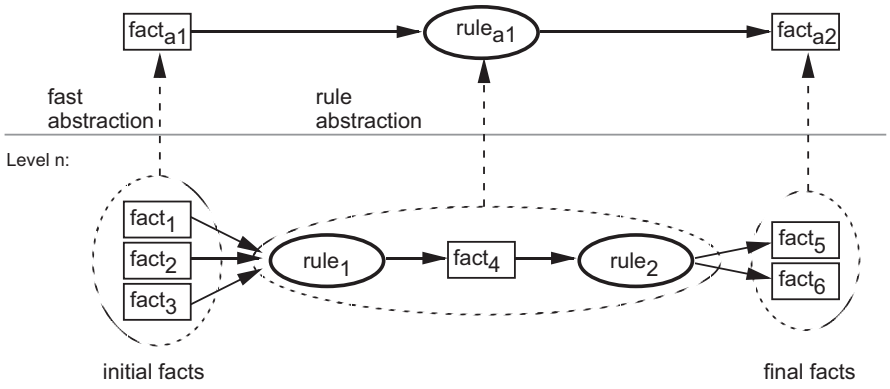

Abbildung 8.4: Mehrstufige Erklärungsstrukturen aus [Bergmann *et al.*, 1994]

Anfragen bestehen aus Ausgangsfakten. Fälle enthalten zusätzlich eine Struktur aus Regeln und Fakten. Der ähnlichste Fall wird über einen Attributvergleich der Ausgangsfakten gefunden. Zunächst beginnt die Suche auf der abstraktesten Ebene. Ist es auf einer Ebene möglich, die Ausgangsfakten der Anfrage denen des Falles über einen Namensvergleich zuzuordnen, so wird die nächst konkretere Ebene verglichen. Das Ergebnis ist wie bei TOPO eine Zuordnung von Teilen der Anfrage zu Teilen des Falles.

Der von [Bergmann *et al.*, 1994] vorgestellte Ansatz ist eine Mischung aus attributbasiertem und strukturorientierten fallbasierten Schließen. Da die Anfragen keine Struktur besitzen, jedoch über eindeutige Namen vergleich-

bar sind, reicht zum Retrieval ein attributbasierter Vergleich der Namen. Die Übertragung und Anpassung ist sehr ähnlich zu der im SFBS. Die Struktur des Falles wird auf die Anfrage übertragen und getestet. Dieser Test entspricht dem Rekonstruktionsschritt, nur daß das Ergebnis keine angepaßte Struktur, sondern eine Aussage über die Anwendbarkeit der Fallstruktur ist. Diese Anpassung wäre auch mit dem Algorithmus von TOPO möglich, wobei jedoch der erste während der Anpassung festgestellte Widerspruch keine Relaxation bewirken würde, sondern ein Verwerfen des Falles.

8.4.2 Resyn: Synthesepläne

Das von [Napoli und Lieber, 1994] entwickelte System Resyn dient der Berechnung von Syntheseplänen zur Erzeugung chemischer Moleküle. Für ein Molekül sucht Resyn nach einem vergleichbaren Molekül, für welches ein Syntheseplan zur Erzeugung bekannt ist. Auf dem gefundenen Syntheseplan aufbauend wird ein Syntheseplan für das Molekül der Anfrage rekonstruiert.

Der Vergleich zweier Moleküle basiert auf dem System bekannten Transformationsregeln und einem Strukturvergleich. Es gibt drei verschiedene Arten von Transformationsregeln: Abstraktion ($\preceq$), Spezialisierung ($\succeq$) und Reduktion ($\Rightarrow$). Jede Transformation verursacht Kosten. Resyn sucht einen Transformationspfad $S[m_k, m]$, der möglichst geringe Kosten verursacht.

Die oberste Zeile in Abbildung 8.5 zeigt ein Beispiel. m ist das Molekül der Anfrage und m_k ein Molekül aus der Fallbasis. Eine Abstraktion $\preceq$ bedeutet umgangssprachlich, daß eine Art reaktiver Kern bestimmt wird. Im Beispiel ist m_2 der reaktive Kern des Moleküls m_k der Fallbasis. m_1 ist eine Spezialisierung dieses Kernes, und somit ist m_2 ebenfalls der reaktive Kern des Moleküls m_1. $m \Leftarrow m_1$ bedeutet, daß m zu m_1 reduziert werden kann. Die Reduktion ist das Inverse der Synthese, also der Erzeugung eines Moleküles aus einem anderen. Mit anderen Worten, m kann aus m_1 erzeugt werden.

Neben den genannten Transformationsregeln kennt Resyn auch eine Distanzfunktion $e(M_1, M_2)$, welche, basierend auf einem Strukturvergleich, die Distanz zweier Moleküle bestimmt. Diese wird für ein A^*- Plansuchverfahren [Pearl, 1984] benutzt, um den billigsten Transformationspfad von m zu m_k zu finden.

Ist ein m_k mit entsprechendem Transformationspfad gefunden, so werden die im Transformationspfad enthalten Transformationen auf den Syntheseplan $P[m_k]$ von m_k angewandt. Abbildung 8.5 zeigt die schrittweise Transformation des Syntheseplans. $P[m]$ ist der Syntheseplan für m, der aus

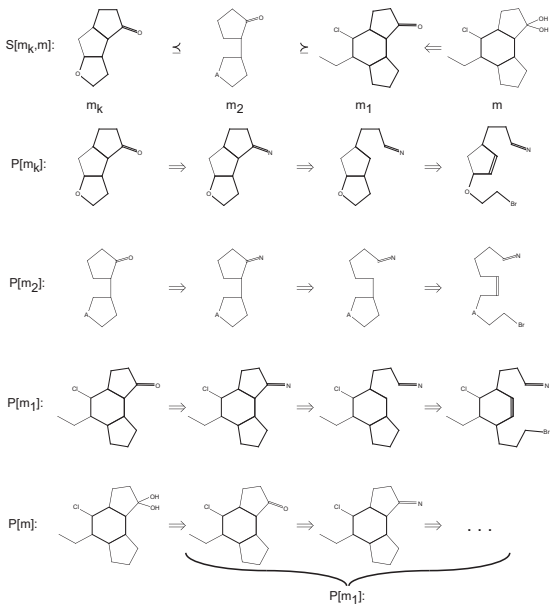

Abbildung 8.5: Pfeile (Reduktionen) sind das Inverse zur Erzeugung und bilden den Syntheseplan. Der Kern ist fett markiert: also $S[m_k, m]$ heißt: erst Abstraktion, dann Spezifikation und dann Erzeugung(aus [Napoli und Lieber, 1994]).

dem Syntheseplan $P[m_1]$ und der aus dem Transformationspfad bekannten Reduktion $m \Rightarrow m_1$ besteht.

Resyn benötigt sehr viel Wissen, bearbeitet jedoch auch eine sehr komplexe Aufgabe. Die Komplexität der A^*-Suche ist ebenfalls sehr hoch, da für jedes Molekül der Fallbasis versucht wird, einen Transformationspfad zu m zu finden. Im Prinzip wäre es auch möglich, den kompletten Syntheseplan unter Benutzung der Reduktionsregeln neu zu berechnen, was jedoch wohl noch aufwendiger wäre.

Verglichen mit dem von mir vorgestellten SFBS benötigt Resyn keine Erkennungsfunktionen, da die Molekülstruktur bekannt ist. Den anwendungsunabhängigen Teil der Funktionalität von Resyn, wie zum Beispiel den Strukturvergleich, könnte TOPO leisten. Resyn besitzt jedoch weitere Funktionalität, wie zum Beispiel das integrierte A^*- Plansuchverfahren, welches die sehr aufwendige und anwendungsspezifische Definition von Ähnlichkeit umsetzt. Diese Suche nach dem kürzesten Transformationspfad ließe sich für TOPO auch als Graphmatching-Problem formulieren, was jedoch angesichts der intuitiven Anwendbarkeit von A^* übertrieben erscheint. Resyn ist folglich ein Beispiel für eine Anwendung, in der die Menge und Art des anwendungsspezifischen Wissens nicht nur spezielles Wissen, sondern auch spezielle Funktionalität erfordert, die nur umständlich in TOPO integrierbar ist.

8.4.3 Composer: Montagepläne

Der Anwendungsbereich von Composer ist die Erstellung von Plänen zur Montage von Motoren und mechanischen Maschinen. Abbildung 8.6 zeigt zum Beispiel eine Anfrage, welche aus vier Objekten besteht. Neben den Objekten beinhaltet eine Anfrage auch Beziehungen (V_i), die das Problem der Anfrage genauer spezifizieren. Der Behälter soll am Ende den Stab enthalten (V_2) und mit Stiel und Deckel verschlossen sein (V_1, V_3).

Fälle enthalten zusätzlich zu den bisher genannten Objekten und Beziehungen Constraints, welche die Reihenfolge der Erfüllung der Beziehungen beschreiben. Zum Beispiel bedeutet das Constraint „$(vor(V_2 < V_1)(V_2 < V_3))$, daß die Beziehung „Stab in Behälter" (V_2) erfüllt werden sollen, bevor der Stiel und der Deckel den Behälter verschließen (V_1, V_3).

Composer benötigt keine Erkennungsfunktionen für die Struktur, da die Struktur in Anfrage und Fällen bereits enthalten ist. Die Vergleichsfunktion wird in [Purvis und Pu, 1995] nicht konkret beschrieben. Sie schreiben jedoch, daß über einen Vergleich der Strukturen die größte gemeinsame Teil-

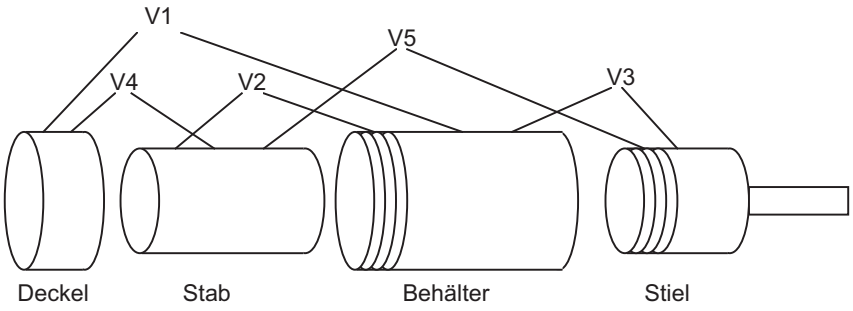

Abbildung 8.6: Eine Anfrage besteht aus Objekten und Beziehungen (aus [Purvis und Pu, 1995]).

struktur in Anfrage und Fall bestimmt wird. Dabei lassen sie auch kontinuierliche Kompatibilitätsfunktionen (vergleiche Abschnitt 4.2.3) zu, um zum Beispiel einen Behälter und einen offenen Zylinder einander zuzuordnen. Das Ergebnis des Vergleichs ist eine Zuordnung von Beziehungen und Objekten aus Anfrage und Fall.

Auf die Anfrage werden auf Basis der Zuordnungen ausschließlich Constraints übertragen. Composer kann auch Constraints mehrerer Fälle auf die Anfrage übertragen, nachdem Teile der Fälle Teilen der Anfrage zugeordnet wurden.

Um den Plan zu erstellen, wird ein Constraint-Relaxations-Algorithmus benutzt, der in der Lage ist, jegliche Konflikte durch anwendungsspezifisches Wissen zu beheben. Konflikte können zum Beispiel dadurch entstehen, daß die Constraints der verschiedener Fälle sich widersprechen, nachdem sie in die Anfrage übertragen wurden.

Das fallbasierte Vorgehen benutzt Composer, um schneller eine Lösung zu finden als ohne Beispiele. Der Anwendungsbereich ist jedoch vollständig modelliert, was auch erklärt, wieso alle Konflikte automatisch behoben werden können. Die verschiedenen Teilschritte passen zu dem Konzept des SFBS bis darauf, daß keine Erkennung der Beziehungen nötig ist. Die über die Funktionalität des SFBS hinausgehende gleichzeitige Verarbeitung mehrerer Fälle ist im allgemeinen nicht möglich, da das Wissen fehlt, um auftretende Konflikte zu beheben.

8.4.4 IDIOM: Topologische und geometrische Anpassung

IDIOM [Smith *et al.*, 1996] wurde ausschließlich zur Anpassung entwickelt. Der als Beispiel gewählte Anwendungsbereich ist der Entwurf von Wohnungen. Der Kern ist ein geschickter Ansatz zur Lösung eines Constraintnetzes.

Abbildung 8.7 zeigt ein Beispiel. Die Eingabe der Anpassung besteht aus einem Modell des Anwendungsbereiches, einer Anfrage, einem Fall und vom Benutzer vorgegebenen Constraints, welche den Fall mit der Anfrage verbinden. Im Beispiel besteht die Anfrage aus drei Zimmern einer Wohnung, und der Fall zeigt ein Schlafzimmer. Der Benutzer gibt als Constraints vor, daß das Schlafzimmer das Wohnzimmer und den Flur berühren soll und daß die obere Wand des Schlafzimmers mit der oberen Wand des Flures abschließen soll. In der Lösung auf der rechten Seite der Abbildung ist die Länge des Flures und des Schlafzimmers angepaßt worden. Daß sowohl der Flur als auch das Schlafzimmer angepaßt wurden, liegt wahrscheinlich an den Constraints des Modells des Anwendungsbereiches oder am Benutzer. Dieses wird in dem Artikel jedoch nicht näher erläutert.

Die zur Anpassung benutzten Constraints werden auf drei Arten vorgegeben: zum einen sind es die Constraints des Modells, zum anderen die vom Benutzer vorgegebenen Constraints, und zum dritten topologische Beziehungen, die in Anfrage und Fall automatisch erkannt wurden. Die verschiedenen Constraints beschreiben die Distanzen zwischen Objekten, Größenverhältnisse und können ausdrücken, daß zwei Grenzen zweier Objekte auf einer orthogonalen Linie liegen. Prinzipiell sind sie also sehr ähnlich zu denen, welche in TOPO verwendet werden.

Zusätzlich besitzen die Constraints jedoch Gewichte, die sich aus dem Modell ableiten oder vom Benutzer vorgegeben werden. Während der Anpassung verdrängen härtere Constraints schwächere. Es findet kein Backtracking statt. Die Besonderheit ist wohl eine sehr geringe Komplexität, welche durch eine sogenannte „Dimensionsreduktion" (siehe [Smith *et al.*, 1996]) erreicht wird.

Verglichen mit dem SFBS fehlen die Vergleichsfunktion, das Matching und eine Übertragungsheuristik. Dies wird damit begründet, daß nur der Benutzer selbst weiß, welchen Teil welchen Falles er an welcher Stelle in die Anfrage integriert haben möchte. Gegen diese Behauptung, daß das System nicht wissen kann, was der Benutzer möchte, kann man nicht argumentieren, da niemand den prototypischen Benutzer kennt. Allerdings kann es Tausende von Fällen geben, die niemand von Hand durchsuchen möchte, und es bestehen aus meiner Sicht gute Chancen, einem Benutzer durch die Funktionalität

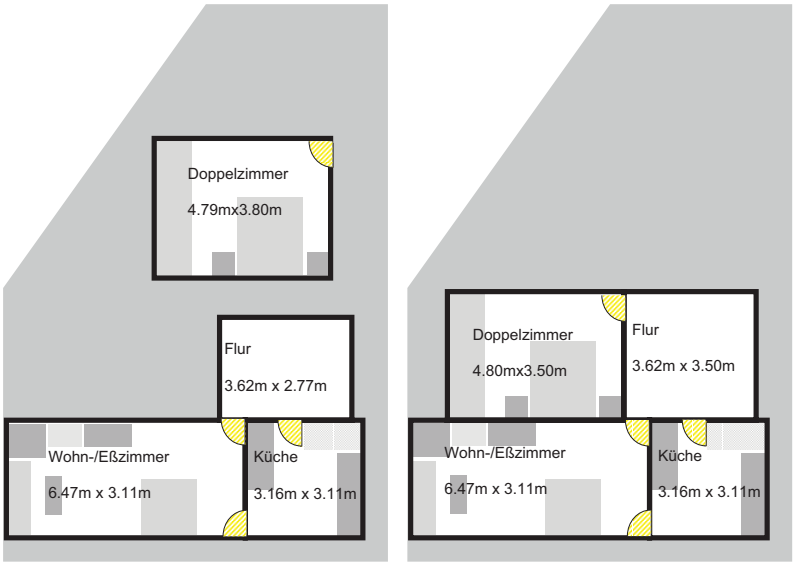

Abbildung 8.7: Dies Beispiel zeigt die Integration eines Doppelzimmers in eine Wohnung. Die graue Fläche im Hintergrund entspricht dem Grundriß des Grundstücks. (Aus [Smith *et al.*, 1996]).

des SFBS dabei zu helfen.

Der Anpassungsschritt ist ähnlich wie bei TOPO gelöst. Bei IDIOM wird jedoch sehr viel Information von der anwendungsspezifischen Wissensbasis oder dem Benutzer gefordert. Das SFBS kommt, wie gezeigt, mit sehr wenig anwendungsspezifischem Wissen aus, da auch die vor der Anpassung durchgeführten Schritte die Struktur berücksichtigen.

Zusammenfassung der Antworten:

1. $\Rightarrow$ Keiner der skizzierten Ansätze kann uneingeschränkte Graphen effizienter vergleichen als in Kapitel 4 gezeigt. Schränkt man jedoch die Graphen auf Baumstrukturen ein, macht Abstriche in der geforderten Qualität oder stellt beliebig Speicherplatz zur Verfügung, so kann die Effizienz des Vergleichs deutlich gesteigert werden.
2. $\Rightarrow$ Im Gegensatz zum SFBS sind andere Ansätze auf isolierte Anwendungsbereiche beschränkt und/oder benötigen viel anwendungsspezifisches Wissen. Außerdem wird dieses Wissen nicht generisch spezifiziert, und die Funktionalität der Ansätze deckt meistens nur einen Bruchteil der Funktionalität des SFBS ab.

Kapitel 9

Zusammenfassung und Ausblick

Alle in dieser Arbeit präsentierten Ideen und Ziele sind durch den Wunsch entstanden, eine konstruktive Anpassung von CAD-Plänen in der Architektur zu ermöglichen. Da die Architektur nicht vollständig modellierbar ist, *muß die Anpassung mit extrem unvollständigem Wissen auskommen. Entstanden ist ein generisches Konzept, daß das zur Anpassung benötigte Wissen für beliebige strukturorientierten Anwendungen genau spezifiziert.* Deshalb beschäftigten sich große Teile dieser Arbeit mit dem benötigten Wissen (Zusammenfassung in Abschnitt 9.1). Ein weiterer Schwerpunkt war der Vergleich von Strukturen, der das zum Retrieval und zur Anpassung benötigte Matching realisiert (Zusammenfassung in Abschnitt 9.2). Durch die erfolgreiche Bearbeitung der beiden ersten Schwerpunkte ergibt sich die eigentliche Anpassung fast von selbst, so daß der entsprechende zusammenfassende Abschnitt 9.3 relativ kurz ist.

Alles zusammen ergibt ein abgeschlossenes Konzept, welches durch seine klare Struktur (Abschnitt 3.3) und seine sehr geringen, genau spezifizierten Anforderungen einfach auf andere Anwendungsbereiche übertragen werden kann (vergleiche Kapitel 7). Abschließend faßt Abschnitt 9.4 die in dieser Arbeit diskutierten möglichen Erweiterungen zusammen.

9.1 Wissen

Das benötigte Wissen wurde in Kapitel 2 am Beispiel vorgestellt, seine Verwendung in Kapitel 3 beschrieben und in Kapitel 7 allgemein spezifiziert und

für weitere Anwendungsbereiche skizziert. Die wichtigsten Punkte waren:

- Das benötigte anwendungsspezifische Wissen unterteilt sich in Erkennungs-, Vergleichs-, Übertragungs- und Rekonstruktionsfunktionen.

- Für die Vergleichsfunktionen und Übertragungsheuristiken gibt es allgemein anwendbares Wissen, da im allgemeinen Probleme gleicher Struktur auf die gleiche Weise gelöst werden können.

- Ist die Anfragestruktur nicht vollständig identisch zu einem Teil der Fallstruktur, sondern nur teilweise ähnlich, so bleibt der Fall anpaßbar, falls die ähnlichen Zusammenhänge durch die Erkennungsfunktionen erkannt und die Auswirkungen der Unterschiede durch die Rekonstruktionsfunktionen berechnet werden können (Abschnitte 2.4 und 7.3).

- Je flächendeckender die Fallbasis ist, desto weniger Unterschiede wird es zwischen Anfrage und ähnlichstem Fall geben, so daß man mit wenig anwendungsspezifischem Regelwissen auskommt (Abschnitt 7.1).

- In einigen Anwendungen wird überhaupt kein anwendungsspezifisches Wissen benötigt (Abschnitt 7.4.1).

- Gegebenenfalls benötigtes Regelwissen ist einfacher zu akquirieren und zu modellieren als bei regelbasierter Problemlösung (Abschnitt 2.4).

- Bewertungskriterien für die Qualität von Anfragen und Ergebnissen können aus der Fallbasis gelernt werden (Abschnitt 3.2).

- Ebenso kann eine Verfeinerung der Übertragungsheuristik gelernt werden (Abschnitt 5.1).

- Beschreiben die Fälle jedoch einzelne Arbeitsschritte, so reicht die allgemeine Übertragungsheuristik aus (Abschnitt 7.1).

- Da die vorgegebenen Funktionen des vorgestellten Konzeptes auf beliebigen Strukturen arbeiten und das anwendungsspezifische Wissen aus Anfragen und Fällen diese Strukturen erzeugt, sind die vorgegeben Funktionen vom Datenformat der Anfragen und Fälle weitgehend unabhängig (Abschnitt 7.2).

9.2 Vergleich

Der Vergleich von Strukturen wurde in Kapitel 4 behandelt. Die Kernaussagen waren:

- Durch Benutzung approximativer Vergleiche zum Retrieval können viele exponentielle Graphvergleiche eingespart und damit auch größere Fallbasen durchsucht werden (Abschnitt 4.3).
- Um dabei jeden einzelnen Vergleich maximal auszunutzen, müssen alle drei Varianten der Dreiecksungleichung benutzt werden, welches durch Erweiterung des Retrieval-Werkzeuges ASPECT möglich ist (Abschnitt 4.3) .
- Die Laufzeit des zumindest zur Anpassung benötigten Graphvergleichs läßt sich durch die Beachtung der 1-Kanten (Abschnitt 4.1.4) und der Verwendung anwendungsspezifischer komplexer Relationen (Abschnitt 4.2.2) reduzieren.
- Die Zuordnung von nicht identischen, aber ähnlichen Relationen und Objekten läßt sich leicht in das vorgestellte Konzept integrieren. Es kann jedoch dabei, unabhängig von dem vorgestellten Verfahren, zu schwer nachvollziehbaren Ergebnissen des Vergleichs kommen. (Abschnitt 4.2.3)
- Da der exponentielle Vergleich nicht vermieden werden kann, werden während des Vergleichs die Abarbeitung des Suchraumes und die bisher erzielten Ergebnisse transparent gemacht, um dem Benutzer die Möglichkeit zu bieten, vorzeitig abzubrechen (Abschnitt 6.4).

9.3 Anpassung

Aufgrund der bisher zusammengefaßten Ergebnisse sind vor Beginn der Anpassung die zu übertragenden Zusammenhänge des Falles und deren Beziehung zur Anfrage bekannt. Entsprechend kurz ist das die Anpassung beschreibende Kapitel 5. Trotzdem gibt es auch bei der Anpassung interessante Aspekte:

- Das vorgestellte Verfahren ermöglicht eine Korrektur der Anfrage (Abschnitt 6.5).

- Bei der Anpassung auftretende Konflikte können teilweise durch Anpassung von Objektattributen gelöst werden (Abschnitt 5.2). Um keine unzulässigen Anpassungen durchzuführen, wird die Anpassung der Attribute durch aus der Fallbasis erzeugte Statistiken (Abschnitt 3.3.1) kontrolliert.

- Mehrere Fälle können nacheinander angepaßt werden (Abschnitt 5.4).

9.4 Ausblick

Die Arbeit beschreibt ein abgeschlossenes generisches Konzept, das auf die Verarbeitung orthogonaler Layouts angewandt wurde. In den verschiedenen Kapiteln wurden jedoch Möglichkeiten aufgezeigt, die Effizienz weiter zu erhöhen, die Ausnutzung der Fallbasis weiter zu verbessern, die Qualität der Anpassung zu steigern und weitere Anwendungsbereiche zu erschliessen:

- Die Verwendung komplexer Relationen würde die Komplexität des Graphvergleichs weiter senken. Bei ihrer Realisierung müßten jedoch die in Abschnitt 4.2.2 diskutierten Probleme gelöst werden.

- Durch die Einführung kontinuierlicher Kompatibilitätsfunktionen könnten Objekte aus Fall und Anfrage einander zugeordnet werden, die nicht identisch, aber ähnlich zueinander sind. Dadurch würden mehr Fälle zur Anfrage passen, so daß die Fälle der Fallbasis besser ausgenutzt werden könnten. Dabei wäre jedoch ein Hauptproblem, daß die Zuordnungen ähnlicher Objekte und Relationen die Anzahl der möglichen Zuordnungen, und damit die Komplexität des Vergleichs, wesentlich erhöhen würde. Zum anderen wäre es unklar, wie die verschiedenen Zuordnungen sinnvoll bewertet werden könnten (vergleiche Abschnitt 4.2.3).

- Wie in Abschnitt 6.6 erwähnt, ist die Anpassung zwar schnell, entspricht jedoch nicht immer dem Benutzerwunsch. Würde man die Benutzerwünsche modellieren, zum Beispiel durch eine Gewichtung der Relationen, müßte man ein geeignetes Constraints-Relaxations-Verfahren suchen oder aber das beschriebene Verfahren erweitern (vergleiche Abschnitt 5.2).

- Die in Abschnitt 7.4.2 vorgestellte Modellierung nicht orthogonaler Layouts zur Verarbeitung von GIS-Plänen ist bisher noch nicht an

realen Daten erprobt. Eine praktische Evaluierung und Verfeinerung der Relationen würde ein weites Anwendungsfeld erschließen.

- Wie oft erwähnt, sind die wesentlichen Punkte des vorgestellten Konzeptes sein generischer Anspruch und seine einfache Anwendbarkeit. Für beides würde ich gerne in möglichst vielen Anwendungsbereichen den praktischen Beweis erbringen.

Anhang A

Lebenslauf

Angaben zur Person

3.6.1968	Geboren in Braunschweig
Eltern:	Dr. Ing. Helmut Coulon, Dipl. Ingenieur
	Lore Coulon geb. Waßmann, Lehrerin
Geschwister:	Dipl. Ing. Carsten Coulon geb. am 2.7.1962
	Dipl. Chem. Cathrin Coulon geb. am 2.11.1963
Familienstand:	verheiratet seit dem 28.5.1993
Ehefrau:	Sabine Coulon geb. Lückenbach, Ärztin
Nationalität:	deutsch
Privatadresse:	Eifelstr. 2, 53119 Bonn, Tel: 0228 / 690763

Ausbildung

1974-1978	Grundschule, Peter-Grieß-Straße, Köln Flittard
1978-1982	Genoveva Gymnasium, Köln Mühlheim
1982-1987	Carl-Duisberg-Gymnasium, Leverkusen
27.6.1987	Abitur
1.10.1987	Einschreibung an der Rheinischen Friedrich-Wilhelms-Universität Bonn, Fach Informatik
Sommer 1987	Vordiplom
30.11.1992	Diplom, Gesamtnote „gut"
	Diplomarbeit: „Wissensbasiertes Zeitmanagement auf der Metaebene", Vertiefungsfach: Künstliche Intelligenz, Nebenfach: Operation Research

Beruflicher Werdegang

Sommer 1990 - 1.12.1992	Mitarbeit als studentische Hilfskraft in der Abteilung FIT.KI der GMD
seit 1.12.1992	Wissenschaftlicher Mitarbeiter in der Abteilung FIT.KI der GMD in den Projekten FABEL (Integration fallbasierter und modellbasierter Methoden) und KIKon (resourcenorientiertes Konfigurieren von technischen Einrichtungen).

Erklärung

Ich erkläre hiermit, daß ich nur die im Literaturverzeichnis angegebenen Hilfsmittel und Quellen verwendet habe.

Veröffentlichungen

[Voß *et al.*, 1992] A. Voß, W. Karbach, C.-H. Coulon, U. Drouven und B. Bartsch-Spörl. Generic specialists in competent behavior. In B. Neumann (Hrsg.), *ECAI 92: 10th European Conference on Artificial Intelligence, Aug. 1992*, Seite 567–571, Wien, 1992. Wiley, Chichester.

[Coulon, 1993] C.-H. Coulon. Image retrieval without recognition. In M. M. Richter, S. Wess, K.-D. Althoff und F. Maurer (Hrsg.), *First European Workshop on Case-Based Reasoning (EWCBR'93), Posters and Presentations*, Band 2, Seite 399–402, Kaiserslautern, Germany, November 1993. Springer, Berlin.

[Coulon *et al.*, 1993a] C.-H. Coulon, H. Dürschke, W. Gräther, B. Linowski und W. Oertel. Fallverwaltung und Fallretrieval: Implementierungen und Tests. Fabel-Report 7, GMD, Sankt Augustin, Februar 1993.

[Coulon *et al.*, 1993b] C.-H. Coulon, F. van Harmelen, W. Karbach und A. Voß. Controlling generate & test in any time. In *Proceedings of GWAI-92*, Band 671 der *Lecture Notes in Artificial Intelligence*, Seite 304–306, Bonn, 1993. Springer, Berlin.

[Dürschke *et al.*, 1993] H. Dürschke, W. Oertel, C.-H. Coulon, W. Gräther, B. Linowski, B. Schmidt-Belz und L. Hovestadt. Der Designkatalog: Ein erster Schritt in Richtung eines FABEL-Anwendungssystems. Fabel-Report 10, GMD, Sankt Augustin, September 1993.

[Voß *et al.*, 1994] A. Voß, C.-H. Coulon, W. Gräther, B. Linowski, J. W. Schaaf, B. Bartsch-Spörl, K. Börner, E.-C. Tammer, H. Dürschke und M. Knauff. Retrieval of similar layouts – about a very hybrid approach in FABEL. In J. S. Gero und F. Sudweeks (Hrsg.), *Artificial Intelligence in Design'94*, Seite 625–640, Lausanne, 1994. Kluwer Academic Publishers, Dordrecht.

[Coulon und Gebhardt, 1994a] C.-H. Coulon und F. Gebhardt. Evaluation of retrieval methods in case-based reasoning. In M. Keane, J.-P. Haton und M. Manago (Hrsg.), *EWCBR-94: Second European Workshop on Case-Based Reasoning*, Seite 283–291, Chantilly, 1994. AcknoSoft Press, Paris.

[Coulon und Gebhardt, 1994b] C.-H. Coulon und F. Gebhardt. Evaluation of retrieval methods in case-based design. Fabel-Report 24, GMD, Sankt Augustin, September 1994.

[Coulon und Steffens, 1994] C.-H. Coulon und R. Steffens. Comparing fragments by their images. In A. Voß (Hrsg.), *Similarity Concepts and Retrieval Methods*, Band 13 der *Fabel-Reports*, Seite 36–44. GMD, Sankt Augustin, 1994.

[Coulon, 1995a] C.-H. Coulon. Automatic indexing, retrieval and reuse of topologies in complex designs. In P. J. Pahl und H. Werner (Hrsg.), *Proceeding of the Sixth International Conference on Computing in Civil and Building Engineering*, Seite 749–754, Berlin, 1995. Balkema, Rotterdam.

[Coulon, 1995b] C.-H. Coulon. Automatic indexing, retrieval and reuse of topologies in architectural layouts. In M. Tan und R. Teh (Hrsg.), *The Global Design Studio – Proceedings of the 6th International Conference on Computer-Aided Architectural Design Futures*, Seite 577–586, Singapore, 1995. Centre for Advanced Studies in Architecture, National University of Singapore.

[Coulon, 1995c] C.-H. Coulon. Benutzung von Datensammlungen zum Layout. In M. Richter und F. Maurer (Hrsg.), *Expertensysteme 95*, Seite 170 – 186, Kaiserslautern, 1995. infix, Sankt Augustin. Proceedings of the German Conference on Expert Systems.

[Coulon, 1995d] C.-H. Coulon. General geometric and topologic retrieval and adaptation (TOPO). In K. Börner (Hrsg.), *Modules for Design Support*, Band 35 der *Fabel-Report*, Seite 33–45. GMD, Sankt Augustin, Juni 1995.

[Coulon, 1996] C.-H. Coulon. Die Rolle des Anpassungswissens im CBR. In H. Czap, P. Jaenecke und H. P. Ohly (Hrsg.), *Analogie in der Wissensrepräsentation: Case-Based Reasoning und räumliche Modelle, 4. Tagung der deutschen Sektion der Internationalen Gesellschaft für Wissensrepräsentation, Okt. 1995, Trier*, Seite 34–41. Indeks Verlag, Frankfurt, 1996.

[Coulon *et al.*, 1996] C.-H. Coulon, W. Gräther, B. Schmidt-Belz, A. Voß, F. Gebhardt, E. Groß und J. W. Schaaf. Virtual building site: supporting building design by multiple methods in FABEL. In J. S. Gero und F. Sudweeks (Hrsg.), *Artificial Intelligence in Design'96*, Seite 465–483, Stanford, 1996. Kluwer Academic Publishers, Dordrecht.

[Voß und Carl-Helmut, 1996] A. Voß und Carl-Helmut. Structural Adaptation with TOPO. In A. Voß (Hrsg.), *Proceedings of AID Workshop on New Directions in Case-based Design*, 1996. To appear.

[Voß und Coulon, 1996] A. Voß und C.-H. Coulon. Structural adaptation with TOPO. In A. Voß (Hrsg.), *ECAI 96, W[orkshop] 6, Adaptation inCase-Based Reasoning*, Seite 52–54. ECAI 96, 1996.

Literaturverzeichnis

[Aamodt und Plaza, 1994] A. Aamodt und E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AICOM*, 7(1):39–59, March 1994.

[Allen, 1983] J. F. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, 26(11):823 – 843, 1983.

[Barrow und Burstall, 1976] H. G. Barrow und R. M. Burstall. Subgraph isomorphism relational structures and maximal cliques. *Information Processing Letters*, 4:83–84, 1976.

[Bender *et al.*, 1990] E. Bender, E. Canfield und B. McKay. The Asymptotic Number of Labeled Connected Graphs with a Given Number of Vertices and Edges. *Random Structures and Algorithms*, 1(2):127 – 169, 1990.

[Bergmann *et al.*, 1994] R. Bergmann, G. Pews und W. Wilke. Explanation-based similarity: a unifying approach for integrating domain knowledge into case-based reasoning for diagnosis and planning tasks. In S. Wess, K.-D. Althoff und M. M. Richter (Hrsg.), *Topics in Case-Based Reasoning: First European Workshop, EWCBR-93, selected papers*, Nummer 837 in Lecture Notes in Artificial Intelligence, Seite 182–196. Springer, Berlin, 1994.

[Börner *et al.*, 1996] K. Börner, E. Pippig, E.-C. Tammer und C.-H. Coulon. Structural Similarity and Adaptation. In I. Smith und B. Faltings (Hrsg.), *EWCBR-96: Third European Workshop on Case-Based Reasoning*, Seite 58–75. Springer, Berlin, 1996.

[Bron und Kerbosch, 1973] C. Bron und J. Kerbosch. Finding all cliques in an undirected graph. *Communications of the ACM*, 16:575–577, 1973.

[Bunke und Messmer, 1994] H. Bunke und B. T. Messmer. Similarity measures for structured representations. In S. Wess, K.-D. Althoff und M. M. Richter (Hrsg.), *Topics in Case-Based Reasoning: First European Workshop, EWCBR-93, selected papers*, Band 837 der *Lecture Notes in Artificial Intelligence*, Seite 106–118. Springer, Berlin, 1994.

[Coulon und Gebhardt, 1994] C.-H. Coulon und F. Gebhardt. Evaluation of retrieval methods in case-based reasoning. In M. Keane, J.-P. Haton und M. Manago (Hrsg.), *EWCBR-94: Second European Workshop on Case-Based Reasoning*, Seite 283–291, Chantilly, 1994. AcknoSoft Press, Paris.

[Coulon, 1993] C.-H. Coulon. Image retrieval without recognition. In M. M. Richter, S. Wess, K.-D. Althoff und F. Maurer (Hrsg.), *First European Workshop on Case-Based Reasoning (EWCBR'93), Posters and Presentations*, Band 2, Seite 399–402, Kaiserslautern, Germany, November 1993. Springer, Berlin.

[Emde, 1996] W. Emde. KIKON. Zwischenbericht, GMD - Forschungszentrum Informationstechnik GmbH, 1996.

[Freksa, 1992a] C. Freksa. Temporal Reasoning based on Semi Intervals. *Artificial Intelligence*, 54:199 – 227, 1992.

[Freksa, 1992b] C. Freksa. Using orientation information for qualitative spatial reasoning. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space - Proceedings of COSIT*, Seite 162–178, 1992.

[Gentner, 1983] D. Gentner. Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, 7:155–170, 1983.

[Guesgen und Hertzberg, 1992] H. Guesgen und J. Hertzberg. *A Perspective of Constraint-Based Reasoning*. Springer-Verlag, Berlin, 1992. ISBN 3-540-55510-2.

[Hall, 1989] R. P. Hall. Computational approaches to analogical reasoning: A comparative analysis. *Journal of Artificial Intelligence*, 39:39–120, 1989.

[Hammond, 1989] K. J. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, Boston, MA, 1989.

[Hernandez, 1992] D. Hernandez. *Qualitative Representation of Spatial Knowledge*. Doktorarbeit, Technische Universität München, Fakultät für Informatik, 1992.

[Hovestadt, 1994] L. Hovestadt. *A4 – Digitales Bauen: Ein Modell für die weitgehende Computerunterstützung von Entwurf, Konstruktion und Betrieb von Gebäuden.* Nummer 120 in Fortschrittsberichte VDI, Reihe 20, Rechnerunterstützte Verfahren. VDI-Verlag, Düsseldorf, 1994.

[Hovestadt, 1995] L. Hovestadt. PM5 – Planungsmodul. Fabel-Report, GMD, Sankt Augustin, 1995. To appear.

[Koch *et al.*, 1996] I. Koch, T. Lengauer und W. E. An Algorithm for Finding Maximal Common Subtopologies in a Set of Protein Structures. *Journal of Computational Biology*, 3:289–306, 1996.

[Kolodner, 1983] J. L. Kolodner. Indexing and retrieval strategies for natural language fact retrieval. *ACM Transactions on Database Systems*, 8:434–464, 1983.

[Kolodner, 1993] J. L. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann, San Mateo, 1993.

[Latecki, 1993] R. Latecki, L. und Roehrig. Orientation and qualitative angle for spatial reasoning. In *Proceedings of IJCAI-93*, Seite 1544–1549, 1993.

[Napoli und Lieber, 1994] A. Napoli und J. Lieber. A first study on case-based planning in organic synthesis. In S. Wess, K.-D. Althoff und M. M. Richter (Hrsg.), *Topics in Case-Based Reasoning: First European Workshop, EWCBR-93, selected papers*, Nummer 837 in Lecture Notes in Artificial Intelligence, Seite 458–469. Springer, Berlin, 1994.

[Papadias *et al.*, 1995] D. Papadias, Y. Theodoridis, T. Sellis und M. Egenhofer. Topological Relations in the World of Minimum Bounding Rectangles: a Study with R-trees. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1995.

[Pearl, 1984] J. Pearl. *Heuristics - Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Publishing Co., Reading, MA, 1984.

[Polya, 1945] G. Polya. *How to solve it.* Princeton University Press, Princeton, NJ, 1945.

[Purvis und Pu, 1995] L. Purvis und P. Pu. Adaptation using constraint satisfaction techniques. In M. Veloso und A. Aamodt (Hrsg.), *Case Based Reasoning Research and Development: First International Conference, ICCBR-95*, Seite 289–300. Springer, Berlin, 1995.

[Schaaf, 1994] J. W. Schaaf. Using gestalten to retrieve cases. In M. Keane, J.-P. Haton und M. Manago (Hrsg.), *EWCBR-94: Second European Workshop on Case-Based Reasoning*, Seite 75–83, Chantilly, 1994. AcknoSoft Press, Paris.

[Schaaf, 1996] J. W. Schaaf. *Über die Suche nach situationsgerechten Fällen im Case-Based Reasoning.* Doktorarbeit, Universität Kaiserslautern, 1996.

[Schank, 1982] R. C. Schank. Reminding and memory organization: an introduction to MOPs. In W. G. Lehnert und M. H. Ringle (Hrsg.), *Strategies for Natural Language Processing*, Seite 455–493. Erlbaum, Hillsdale, NJ, 1982.

[Slade, 1991] S. Slade. Case-based reasoning: a research paradigm. *AI Magazine*, 12(1):42–55, 1991.

[Smith *et al.*, 1996] I. Smith, R. Stalker und C. Lottaz. Creating Design Objects from Cases for Interactive Spatial Composition. In J. Gero und F. Sudweeks (Hrsg.), *Proceedings of AID Workshop on New Directions in Case-based Design*, Seite 97 – 116, 1996.

[Steele, 1990] G. L. Steele. *Common Lisp: the Language.* Digital Press, Bedford, MA, 2.. Auflage, 1990.

[Steinmann, 1994] F. Steinmann. Wissensbasierte Computerstützung früher Phasen des architektonischen Entwurfs. In *Computer und Architektur*, Band 4, Seite 83 – 97. Hochschule für Architektur und Bauwesen Weimar, 1994. ISBN 0863-0712.

[Tammer *et al.*, 1995] E.-C. Tammer, K. Steinhöfel, S. Schönherr und D. Matuschek. Anwendung des Konzeptes der Strukturellen Ähnlichkeit zum Fallvergleich mittels Term- und Graph-Repräsentationen. Fabel-Report 38, GMD, Sankt Augustin, September 1995.

[Voß, 1996] A. Voß. Towards a methodology for case adaptation. In W. Wahlster (Hrsg.), *ECAI'96, 12th European Conference on Artificial Intelligence, Aug. 1996, Budapest*, Seite 147–151. John Wiley and Sons, Chichester, August 1996.