Automatische Stundenplangenerierung mittels Constraint Logic Programming und der Expertensystementwicklungsumgebung *ECLiPSe*

Diplomarbeit an der Fachhochschule Wedel im WS1999/2000

Referenten : Prof.Dr. Uwe Schmidt (FH-Wedel)

Co-Referent : Prof.Dr. Kurt Brinkmann (FH-Wedel)

Bearbeiter : André Rieck (ii7073 FH-Wedel)

EMail-Adresse des Autors : andre.rieck@gmx.de

Zusammenfassung

Die Erstellung von Stundenplänen für Hochschulen (hier: die Fachhochschule Wedel) stellt ein Problem dar, dessen manuelle Lösung sehr schwierig und zeitaufwendig ist. In der Vergangenheit wurden viele Ansätze untersucht, um dieses Planungsproblem zu lösen. Insbesondere eignet sich der noch recht neue Ansatz des Constraint-Logic-Programming für die Lösung von Planungsproblemen, bei denen sich die Lösung in Form von Randbedingungen (Einschränkungen) formal beschreiben lässt. Diese Diplomarbeit befasst sich mit der Entwicklung eines Systems zur automatischen Generierung von Stundenplänen für die Fachhochschule Wedel mit Hilfe der CLP-Entwicklungsumgebung *ECLiPSe*.

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund der Arbeit

Das Problem der Erstellung von Zeitplänen, daß sich auch als Problem der wechselseitigen Zuweisung von Ressourcen über der Zeit verstehen läßt, tritt in vielen Variationen im Alltag auf. Beispiele sind die Erstellung von Stundenplänen für allgemeinbildende Schulen und Hochschulen, die Erstellung von Klausurplänen in einem begrenzten Zeitintervall für Hochschulen und die zeitliche und räumliche Verteilung von Seminarvorträgen, so daß angemeldete Teilnehmer an möglichst allen Seminarvorträgen teilnehmen können. Die Erstellung solcher Zeit- bzw. Stundenpläne ist eine sehr komplexe und zeitaufwendige Aufgabe, zumal diverse Nebenbedingungen wie Ressourcenanforderungen von Vorlesungen, Seminarvorträgen, etc. und Zeitbeschränkungen eingehalten werden müssen. Solche Zeit bzw. Stundenpläne sind im allgemeinen in regelmäßigen Abständen unter neuen Voraussetzungen erneut zu erstellen, so daß eine automatisierte Stundenplanerstellung lohnend wäre. Auch an der FH-Wedel besteht das Problem der Erstellung eines Stundenplanes, das momentan manuell mit einem Zeitaufwand von etwa 4 Wochen gelöst wird. Um die Größe des Problems darzustellen werden hier einige Eckwerte des zu erstellenden Stundenplanes angegeben.

Vorlesungen etwa 170. Eine Vorlesung kann dabei aus mehreren Blöcken bestehen, die sich unter Umständen über mehrere Zeiteinheiten erstrecken.

Teilnehmergruppen zur Zeit 63.

Dozenten etwa 50.

Zeiten 6 Tage und 8 Vorlesungseinheiten pro Tag (incl. Mittagspause).

Räume 6 Hörsäle, 4 allgemeine Rechenzentren, 4 Seminarräume und diverse Labore/Rechenzentren mit spezieller Ausstattung.

Aufgrund des großen Zeitaufwandes besteht der Wunsch nach einer automatischen Generierung des Stundenplanes mit den o.g. Eckwerten. Diese Diplomarbeit befaßt sich mit der Erstellung eines Systems zur automatischen Generierung eines solchen Stundenplanes.

1.2 Zielsetzung

Ziel war es ein System zu erstellen, daß aus den Eingabedaten über ...

- zu haltende Vorlesungen
  - Teilnehmergruppen und Dozenten
  - zur Verfügung stehende Räume
  - einzuhaltende Nebenbedingungen

einen Stundenplan erstellt, der folgende Anforderungen erfüllt

1. ***Konfliktfreiheit*** Es darf nicht mehr als eine Vorlesung zu einem Zeitpunkt in einem Raum stattfinden. Niemand darf an mehreren Veranstaltungen gleichzeitig teilnehmen müssen.
2. ***Erfüllung von Resourcenanforderungen*** Die Vorlesungen müssen in Räumen stattfinden deren Größe für die Gesamtteilnehmerzahl der Vorlesung ausreichend ist, und die alle während der Vorlesung benötigten Ressourcen (wie zum Beispiel Videoprojektoren etc.) bereitstellen.
3. ***Mittagsfreistunde für alle Teilnehmergruppen*** Alle Teilnehmergruppen müssen zur Mittagszeit mindestens eine freie Vorlesungseinheit haben, um die Möglichkeit zu haben zu Mittag zu essen.
4. ***Einhaltung von speziellen Nebenbedingungen*** Die für spezielle Vorlesungen und Dozenten definierten zeitlichen Einschränkungen (z.b.: Dozent X steht Montag Nachmittags grundsätzlich nicht zur Verfügung) müssen eingehalten werden.
5. ***Vollständigkeit*** Nach Möglichkeit sollen alle definierten Vorlesungen im Stundenplan berücksichtigt werden.

Bevor ein Stundenplan erstellt werden kann müssen die Daten über die zu haltenden Vorlesungen, die teilnehmenden Gruppen, die die Vorlesungen haltenden Dozenten, über zur Verfügung stehende Räume und einzuhaltende Nebenbedingungen etc. gesammelt werden. Die Sammlung und Verifizierung dieser Daten sowie deren Aufbereitung für das Stundenplansystem sind nicht Teil der Diplomarbeit.

1.3 Übersicht

Hier soll eine Übersicht über die einzelnen Teile dieser Ausarbeitung gegeben werden.

In Abschnitt 2 Grundlagen (Seite 6) wird das Stundenplanproblem, einige Probleminstanzen und schon bestehende Lösungsansätze beschrieben, daß Paradigma der Constraint Logischen Programmierung (CLP) erläutert und das als Basis zur Implementierung verwendete CLP-System *ECLiPSe* vorgestellt.

Abschnitt 3 Anforderungsanalyse (Seite 21) beschreibt die Anforderungen an das zu erstellende Stundenplansystem. Insbesondere welche Nebenbedingungen in jedem Fall einzuhalten sind und welche Nebenbedingungen vom Benutzer des Systems für den zu erstellenden Stundenplan definierbar sein sollen.

Im Abschnitt 4 Entwurf (Seite 37) wird das Konzept zur Realisierung eines solchen Systems beschrieben.

Im Abschnitt 5 Implementierung (Seite 50) wird der Programmablauf und der Aufbau interner Datenstrukturen des Stundenplanungssystems beschrieben.

Erweiterungs und Verbesserungsmöglichkeiten werden im Abschnitt 6 Bewertung und Ausblick (Seite 54) dargestellt.

2 Grundlagen

2.1 Das Stundenplanproblem

Abstrahiert man vom konkreten Problem einen Stundenplan für eine Hochschule automatisch zu generieren, so kann das Stundenplanproblem als eine spezielle Form eines allgemeinen Scheduling-Problems betrachtet werden, bei dem es darum geht Prozesse die einen definierten Resourcenbedarf haben und unter Umständen nicht in beliebiger Reihenfolge ablaufen können, so in der Zeit zu verteilen, daß alle Prozesse mit ihren benötigten Resourcen ablaufen können. Unter Umständen müssen dabei bestimmte Optimalitätskriterien erreicht werden. So könnte z.b. die Gesamtdauer aller Prozesse a priori beschränkt sein, oder eine möglichst gute Ressourcenausnutzung gefordert werden.

Die Erstellung von Stundenplänen ist nun eine spezielle Form des Scheduling Problems, bei dem es darum geht eine vordefinierte Menge von Vorlesungen mit definierter Dauer (Prozesse) so innerhalb eines festgelegten Zeitraumes (z.b. einer Vorlesungswoche) auf Zeiten und Räume zu verteilen, daß der Ressourcenbedarf der Vorlesungen sichergestellt, und der so entstandene Stundenplan bezüglich einer Menge von Nebenbedingungen (Constraints) konsistent ist. Der Resourcenbedarf von Vorlesungen läßt sich in den Resourcenbdarf bezüglich der Räume, sowie der Teilnehmer einteilen.

1. ***Resourcenbedarf bezüglich der Räume*** Zum Beispiel: Raumgröße, Vorhandensein von Medien und technischer Einrichtungen, bestimmte Eigenschaften von Räumen (z.b. Rollstuhlgerecht, für Seminare geeignet)

2. ***Resourcenbedarf bezüglich der Teilnehmer*** Auch die Teilnehmer einer Vorlesung werden als Resourcen angesehen, die der Vorlesung zur Verfügung stehen müssen. Teilnehmer einer Vorlesung sind die die Vorlesung haltenden Dozenten/Assistenten und die die Vorlesung hörenden Studenten.

Eine andere Sichtweise desselben Problems ist es den Vorgang der Stundenplanung als das Festlegen von "Treffen" (den Vorlesungen) zwischen Vorlesenden und Hörenden so innerhalb eines festen Zeitraumes zu organisieren, daß eine Menge von Nebenbedingungen erfüllt bleibt.

Die Menge der zu erfüllenden Constraints läßt sich in sogenannte *harte* und *weiche Constraints* sowie in *allgemeine* und *spezielle Constraints* bezüglich des Stundenplanes einteilen. Harte Constraints sind Constraints die erfüllt sein müssen, damit der betrachtete Stundenplan gültig ist. Im Gegensatz dazu sind weiche Constraints solche die möglichst weitgehend erfüllt ein sollten, damit der betrachtete Stundenplan nicht nur gültig, sondern auch qualitativ hochwertig bezüglich der durch die weichen Constraints definierten Optimierungskriterien ist. Allgemeine und spezielle Constraints können jeweils hart als auch weich sein, wobei allgemeine Constraints solche sind, die für jeden zu erstellenden Stundenplan gelten (z.b. Keine Überschneidung von Vorlesungen mit nicht disjunkten Hörergruppen) und spezielle Constraints im Rahmen der Definition eines konkreten zu erstellenden Stundenplanes definiert werden (z.b.: Dozent X hat mindestens einen freien Tag pro Woche und hält nach 15 Uhr keine Vorlesungen).

Je nachdem welches konkrete Stundenplanungssystem betrachtet wird, unterscheiden sich die allgemeinen und spezifischen Constraints. So kann es zum Beispiel in einem Fall notwendig sein einen Stundenplan zu erstellen in dem sich Vorlesungen mit gemeinsamen (nicht disjunkten) Hörergruppen niemals überschneiden, während in einem anderen Fall nur die Minimierung von solchen Überschneidungen gefordert wird.

2.1.1 Ähnliche Probleme

Um einen Eindruck zu vermitteln welche ähnlichen Probleme bestehen, für die ebenfalls schon Lösungen bekannt sind folgen hier drei Beispiele für ähnliche Probleme zum Problem des Erstellens eines Stundenplanes für eine Hochschule [CWIR95].

- ***Stundenplanung für allgemeinbildende Schulen***. Wobei die meisten Veranstaltungen in, den Gruppen fest zugeordneten, Räumen stattfinden und auch nur jeweils eine Gruppe von Hörern (Schülern) an einer Veranstaltung (Schulstunde) teilnimmt. Ein sehr wichtiges Optimalitätskriterium wird die Minimierung von Freistunden zwischen den Schulstunden sein. Veranstaltungen mit derselben Teilnehmergruppe (Klasse) dürfen sich keinesfalls überschneiden.
- ***Erstellung von Klausurplänen***. Im Gegensatz zu Stundenplänen dürfen hier möglicherweise mehrere Klausuren gleichzeitig in einem Raum abgehalten werden. Überschneidungen zwischen den Teilnehmergruppen dürfen keinesfalls auftreten, da sonst Teilnehmer mehrere Klausuren gleichzeitig schreiben müssten. Klausurteilnehmer sollten nach Möglichkeit nicht mehrere Klausuren an einem Tag schreiben müssen, und der zeitliche Abstand zwischen zwei Klausuren sollte für möglichst alle Teilnehmer möglichst groß sein.[ASAP]
- ***Erstellung von Konferenzplänen***. Für Konferenzen melden sich Teilnehmer meist unter Angabe der Seminare und Workshops an, an denen sie teilnehmen möchten. Konferenzpläne sollten es den Teilnehmern ermöglichen an möglichst vielen Seminaren und Workshops teilzunehmen. Die Überschneidung von Veranstaltungen mit nicht disjunkten Teilnehmermengen sollte also minimiert werden. Im Gegensatz zu Stundenplänen kann auch die Abfolge von Veranstaltungen von Bedeutung sein, da Workshops und Seminare u.U. aufeinander aufbauen.

Die Anforderungen an einen Stundenplan sind aber auch zwischen Hochschulen so unterschiedlich, daß die existierenden Systeme jeweils für eine bestimmte Hochschule speziell entwickelt wurden, und allgemeinverwendbare Systeme schwer vorstellbar erscheinen.

2.1.2 Durchführbarkeit, Optimalität

In der Literatur wird von Stundenplänen, durchführbaren sowie optimalen Stundenplänen gesprochen.

> A ***timetable***, or schedule is a description of the movement and grouping of resources over time, often to achieve a certain aim or aims and/or subject to a set of constraints. (Literaturangabe)[ASAP]
>
> A ***feasible Timetable***, or schedule, is one that satisfies is associated set of constraints. [ASAP]

Ein Stundenplan ist also durchführbar (feasible) wenn er alle definierten Constraints erfüllt. Mit Constraints sind hier harte Constraints gemeint. Sind zusätzlich noch weiche Constraints bzw. Optimalitätskriterien definiert, so ist ein ***optimaler*** Stundenplan ein durchführbarer Stundenplan der die Erfüllung der weichen Constraints bzw. Optimalitätskriterien maximiert, bzw. die Kosten des erzeugten Stundenplanes minimiert.

Da ein nicht durchführbarer Stundenplan, also zum Beispiel einer nach dem ein Dozent mehrere Veranstaltungen gleichzeitig halten muss, keinen praktischen Sinn macht, müssen alle erzeugten Stundenpläne mindestens durchführbar sein, d.h. alle harten Constraints erfüllen. Viele in der Literatur [CWIR95] angegebene Systeme definieren auch Optimalitätskriterien als harte Constraints, so daß hier generierte durchführbare Stundenpläne auch zu einem gewissen Grad optimal sind.

Die Anforderungen die an die Durchführbarkeit und Qualität von Stundenplänen gestellt werden, variieren stark mit der Einrichtung für die der Stundenplan erstellt wird. Im Folgenden wird eine Auswahl von harten und weichen Constraints aus verschiedenen Systembeschreibungen aufgeführt, um einen Eindruck zu vermitteln, wie sich die Anforderungen an Stundenpläne unterscheiden. Allen Anforderungen ist gemeinsam daß Dozenten jeweils nur eine Vorlesung zur Zeit halten, und daß in einem Raum nur maximal eine Veranstaltung zur Zeit stattfinden kann.

Fachbereich - Informatik - Universität Lissabon [TCLP]:

- Theoretische Kurse dürfen nicht nach 18:00 Uhr stattfinden (H)
- Keine Vernstaltungen nach 20:00 Uhr (H)
- An einem Nachmittag müssen alle Dozenten vorlesungsfrei sein, damit eine wöchentliche Konferenz stattfinden kann (H)
- Alle Studenten müssen zur Mittagszeit eine freie Stunde haben (H)

- Möglichst keine Überschneidung von Veranstaltungen mit nicht disjunkten Hörergruppen (S)

Fachbereich Informatik - Ludwig Maximilians Universität München [CHRT]:

- Zwischen zwei Veranstaltungen eines Dozenten muß mindestens eine Freistunde sein. (H)
- Veranstaltungen eines Semesters des Grundstudiums dürfen sich nicht überschneiden (H)
- Veranstaltungen eines Semesters des Hauptstudiums sollten sich nicht überschneiden (S)
- Zwischen zwei Veranstaltungen einer Vorlesung sollte mindestens ein Tag liegen (S)

FH-Wedel

- Studenten müssen in der Mittagszeit (Vorlesungen von 11:00 bis 13:45) mindestens eine freie Vorlesungseinheit haben (H)
- Die Mittagspause sowie der Samstag sollten nach Möglichkeit ausgespart bleiben (S)
- Veranstaltungen mit nicht nicht disjunkten Hörergruppen dürfen sich nicht überschneiden (H)

(H)=Harte Einschränkung, (S)=Weiche Einschränkung.

Alle Systeme berücksichtigen zusätzliche spezielle harte und/oder weiche Constraints die sich auf Dozenten oder Veranstaltungen beziehen. Wie z.b. "Dozent X hat mindestens N Tage in der Woche Vorlesungsfrei", "Veranstaltung Y darf nicht nach 15:00 Uhr stattfinden" .

2.1.3 Komplexität des Problems

Die Menge aller möglichen, auch nicht durchführbaren, Stundenpläne ist der Suchraum des Stundenplanproblems. Das Aufsuchen eines Punktes im Suchraum der einen durchführbaren oder sogar optimalen Stundenplan darstellt, entspricht der Lösung des Problems. Da das Stundenplanproblem zur Klasse der NP-vollständigen Probleme gehört ([ASAP],[CWIR95]) kann davon ausgegangen werden, daß kein deterministischer Algorithmus zur Lösung dieses Problems existiert. Zur Lösung des Problems muß also im Suchraum nach einer Lösung gesucht werden. Einen Eindruck von der Komplexität des Problems läßt sich durch die Größe des Suchraumes gewinnen. Die Größe des Suchraumes eines Stundenplanes mit $v = 6$ Vorlesungseinheiten (Zeiteinheiten) pro Tag, $t = 5$ Vorlesungstagen pro Woche, $r = 8$ zur Verfügung stehenden Räumen, einer konstanten Vorlesungsdauer für alle Vorlesungen von 1, und 100 zu haltenden Vorlesungen hat folgende Größe :

$$trv^{100} = 240^{100}$$

Es ist also offensichtlich daß das bloße Durchprobieren aller möglichen Kombinationen kein brauchbarer Ansatz wäre, da z.b die Zeit t die für das Absuchen selbst für 1/10 des Suchraumes und einer Zeitspanne von 1nS für das Aufsuchen eines Punktes im Suchraum

$$t[sekunden] = 240^{100} \cdot 10^{-1} \cdot 1 \cdot 10^{-9}s = 240^{90}s$$

also etwa $6.8 \cdot 10^{88}$ Stunden dauern würde.

Der Suchraum muss also nach möglichkeit a priori und während der Suche soweit wie möglich eingeschränkt, und die Suche durch eine geeignete Heuristik geleitet werden. Insbesondere das Einschränken des Suchraums und die frühzeitige Erkennung von Unterräumen des Suchraumes in denen keine Lösung existieren kann die die definierten Einschränkungen erfüllt, ist mit Hilfe von Constraint-Logic-Programming gut möglich.

2.1.4 Lösungsansätze

Hier werden kurz einige Lösungsansätze beschrieben, die schon bei konkreten Umsetzungen der Lösung des Stundenplanproblems verwendet wurden [CWIR95]. Im Anschluß werde ich darstellen welche Vorteile der Ansatz des Constraint-Logic-Programming für die Lösung des Stundenplanungsproblems bietet, und wie er mit anderen erwähnten Ansätzen kombiniert werden kann.

- ***Graph Coloring.*** Bei diesem Ansatz wird das Problem der Studenplanung auf das Problem des "Einfärbens" eines Graphen abgebildet, so daß keine durch Kanten verbundene Knoten dieselbe Farbe haben. Jeder zu haltenden Vorlesung wird dabei ein Knoten im Graphen zugeordnet. Die Vorlesungen die nicht parallel ablaufen dürfen werden mit Kanten verbunden. Die zur Verfügung stehenden Farben entsprechen den zur Verfügung stehenden Zeiteinheiten einer Vorlesungsperiode (also z.b. die Tageszeiten jedes Tages einer Woche). Dieser Ansatz ist schon etwas älter, und wurde häufig verwendet da das zugrundeliegende Problem des "Einfärbens" von Graphen schon gut untersucht war.

- ***Genetische Algorithmen.*** Genetische Algorithmen versuchen das Evolutionsprinzip der Biologie auf die Lösung kombinatorischer Probleme anzuwenden. Individuelle Lösungen eines Problems werden als Individuen angesehen deren "Fittness" (Lösungsgüte) über eine Fittnes-Funktion (Inverse Kostenfunktion) bewertet wird. Dargestellt werden die Lösungen als Strings über einem endlichen Alphabet. Von Generation zu Generation darf sich nur eine bestimmte Menge aus der Population fortpflanzen, wobei die Wahrscheinlichkeit der Fortpflanzung für ein Idividuum mit seiner Fittness zunimmt. Nachdem eine gerade Anzahl von Individuen zur Fortpflanzung ausgewählt wurde, werden Individuenpaare gebildet, und für jedes Paar ein sogenannter "Crossover Point" gewählt, der als Index in den Individuenstrings dient. Für jedes Paar wird dann ein Nachfolgeindividuum gebildet, indem der Substring bis zum "Crossover Point" des einen "Elternteils" mit dem Substring ab dem "Crossover Point" des anderen Elternteils konkateniert wird. Zusätlich kann jedes Element des resultierenden Lösungsstrings mit einer kleinen Wahrscheinlichkeit zufällig abgeändert (mutiert) werden. Der Evolutionszyklus kann wiedeholt werden bis eine aktzeptable Lösung gefunden oder eine maximale Obergrenze der Anzahl der Evolutionszyklen erreicht wurde.

- ***Direkte Heuristiken.*** Hierbei wird versucht das Lösungsverhalten von Menschen auf die automatische Stundenplangenerierung zu übertragen, indem der Stundenplan Vorlesung für Vorlesung aufgebaut wird, und dabei die am stärksten beschränkten Vorlesungen zuerst festgelegt werden.

- ***Lokale Suchverfahren.*** Sind Optimierungsverfahren, die sich ausgehend von einem Punkt im Suchraum des Problems in der loklen "Nachbarschaft" dieses Punktes von Punkt zu Punkt bewegen, und dabei versuchen die Kosten der Lösung zu minimieren. Typisch für lokale Suchverfahren ist, daß die Suche nicht über partielle Lösungen des Problems, also über Unterräume des Suchraums, zur Lösung findet, sondern sich nur von Punkt zu Punkt bewegt [STUT99]. Lokale

Suchverfahren sind z.b.

- *Hill Climbing.* Hierbei wird immer der Punkt als Nachfolger in der Suche ausgewählt, der die geringsten Kosten, bzw. den höchsten Gewinn bringt. Der Nachteil dieser Art der lokalen Suche ist, daß sich der Suchprozess in lokalen Maxima (Bezogen auf den Gewinn) festsetzen kann.
- *Simulated Annealing.* Dieser Suchalgorithmus verhält sich ähnlich wie das Hill Climbing, jedoch akzeptiert er mit einer gewissen Wahrscheinlichkeit, die mit der Zahl der Iterationen sinkt, auch einen Anstieg der Kosten. So soll ein frühzeitiges Festlaufen des Algorithmus in lokalen Maxima vermieden werden.

∘ ***Constraint-Logic-Programming.*** Hierbei wird die mögliche Lösung über Einschränkungen (Constraints) definiert. Diese Einschränkungen beschneiden a priori den Suchraum. Werden während der Lösungssuche beschränkten Problemvariablen Werte zugewiesen, so lassen sich über die definierten Constraints evtl. rekursiv Einschränkungen der Wertebereiche anderer Problemvariablen ableiten (Constraint Propagation), so daß das System der definierten Constraints konsistent bleibt. Auch "Sackgassen" im Suchbaum, die zu keiner konsistenten Lösung mehr führen können, lassen sich durch Constraint Propagation frühzeitig erkennen.

Die oben erwähnten Ansätze sind nicht speziell für das Stundenplanungsproblem entwickelt worden, und stammen mit Ausnahme des Graph-Coloring aus dem Gebiet der Künstlichen Intelligenz.

Das Graph-Coloring ist der älteste der oben beschriebenen Ansätze, und war für das Stundenplanungsproblem gut geeingnet, da die zugrundeliegenden Algorithmen schon gut untersucht waren, und die anderen Ansätze mit Ausnahme der direkten Heuristiken noch nicht oder nicht ausreichend untersucht bzw. bekannt waren. Viele spezifische Constraints (z.b. Bestimmte sequentielle Abfolge von Vorlesungen) lassen sich mit diesem Ansatz aber nur schwer realisieren. Es stehen mittlerweile leistungsfähigere und flexiblere Ansätze, insbesondere das Constraint-Logic-Programming in Verbindung mit lokalen Suchverfahren, zur Verfügung, so daß der Ansatz des Graph-Coloring als nicht mehr dem Stand der Technik entsprechend angesehen werden kann.

In der Literatur werden erfolgreiche Problemlösungen des Stundenplanungsproblems mittels Genetischer-Algorithmen recht oft erwähnt (z.b.: [ASAP],[CWIR95]).

Die Implementierung eines Genetischen-Algorithmus für das Stundenplanungsproblem, der auch Wissen über die Struktur des Problems z.b. in Form geeigneter Heuristiken zur Auswahl von zu mutierenden Elementen des Lösungsstring integriert, halte ich für weniger erfolgversprechend. Eine Lösung mittels Genetischer-Algorithmen ähnelt stärker einer zufallsbasierten Bestensuche, und bietet keine guten Möglichkeiten Problemspezifisches Wissen in die Lösung zu integrieren. Ein weiterer Nachteil dieses Ansatzes gegenüber dem CLP ist die Notwendigkeit einer expliziten Implementierung der Fittness-Funktion, die eine Aussage über Güte und Konsistenz der Lösungen machen sollte. Beim CLP-Ansatz wird mindestens die Konsistenz der Lösung vom Laufzeitsystem über die ohnehin definierten Constraints überwacht.

Direkte Heuristiken stellen den Versuch dar eine direkte Lösung des Stundenplanungsproblems ohne einen Suchprozess zu realisieren, indem das Menschliche Verhalten beim Erstellen von Stundenplänen simuliert wird, indem die am meisten eingeschränkten Vorlesungen zuerst im Stundenplan verteilt werden. Dieser Ansatz wird bei komplexeren Probleminstanzen des Stundenplanungsproblems kaum zu einer Lösung führen, und wurde als eigenständiger Ansatz nur in den frühen Lösungsversuchen eingesetzt. In Kombination mit dem CLP und lokaler Suche eignet sich dieser Ansatz jedoch recht gut um zu einer ersten Lösung zu kommen, die u.U. noch keine weichen Constraints berücksichtigt, und später in einem zweiten Lösungsschritt von Lokalen-Suchalgorithmen "repariert" (d.h. optimiert) wird.

Viele der neueren Lösungen bauen auf lokalen Suchverfahren auf. Diese Suchverfahren gehen jedoch von einer schon bestehenden Lösung aus, und versuchen dann diese zu optimieren, wobei die erste Lösung trivial sein kann und nicht durchführbar sein muss. Auch bei den lokalen Suchverfahren muss die Überprüfung der Durchfürbarkeit des Stundenplanes, bzw. der Konsistenz der Lösung mit den harten Nebenbdingungen explizit implementiert werden. Dies geschieht beim CLP-Ansatz implizit durch die Definition des Problems mittels Constraints. Ein weiterer großer Vorteil des CLP Ansatzes gegenüber dem der lokalen Suche und allen anderen erwähnten Verfahren ist die frühzeitige, und während der Lösungssuche fortschreitende Einschränkung des Suchraumes, sowie die mögliche frühzeitige Erkennung von "Sackgassen", durch die Constraint-Propagation Fhähigkeiten der CLP Laufzeitumgebung.

2.1.5 Das konkrete Stundenplanungsproblem an der Fachhochschule Wedel

Wie weiter oben schon beschrieben, finden an der FH-Wedel pro Woche etwa 170 Vorlesungen statt, die von 50 zum Teil externen (nicht an der FH-Wedel fest angestellten) Dozenten gehalten werden. Es existieren momentan 63 Hörergruppen, die sich aus den Fachrichtungen und Fachsemestern ergeben. An 6 Tagen in der Woche (Montag bis Samstag) können Vorlesungen stattfinden, wobei an jedem Tag 8 Zeiteinheiten konstanter Länge zur Verfügung stehen.

Weiterhin kann es notwendig sein Gruppen in Untergruppen aufzuteilen, die dann an parallel stattfindenden Veranstaltungen teilnehmen können.

Beispiel Im Hauptstudium können einige Fachrichtungen ihr Vertiefungsgebiet wählen. Wird z.b. für die Gruppe ii6 eine Vorlesung Hardware-Design gehalten, an der jedoch nur die Vertiefungsrichtung Hardware teilnimmt, und eine Vorlesung Software-Design, an der jedoch nur die Vertiefungsrichtung Software teilnimmt, so würde die Eintragung der Gruppe ii6 in beide Vorlesungen zur Folge haben, daß die Vorlesungen keinesfalls parallel stattfinden könnten, obwohl dies möglich wäre. In diesem Fall muß die Möglichkeit bestehen eine Gruppe (hier ii6) in Untergruppen zu unterteilen, die dann zueinander parallel, nicht jedoch parallel zur Obergruppe (ii6) stattfinden dürfen.

Auf jeden Fall sind folgende Constraints für einen durchführbaren Stundenplan zu erfüllen.

- Keine Überschneidungen von Vorlesungen mit nicht disjukten Teilnehmergruppen
- Nicht mehr als eine Veranstaltung pro Zeit in einem Raum
- Jede Hörergruppe muß die Möglichkeit haben am Mittagessen teilzunehmen, also in der entsprechenden Zeit mindestens eine Vorlesungsfreie Zeiteinheit im Stundenplan haben.
- Einzelne Zeiteinheiten von bestimmten Vorlesungen die aus mehreren Zeiteinheiten bestehen, müssen in zeitlichem sowie räumlichen Zusammenhang stehen (Blockbildung).

Die Gütekriterien sind :

- Je mehr Vorlesungen in die späten Abendstunden, in die Mittagspause (Kern der Mittagszeit) und auf den Samstag fallen, desto schlechter ist der Stundenplan.
- Je mehr spezielle Anforderungen bezüglich der zeitlichen Verfügbarkeit von Dozenten, bzw. der zeitlichen und/oder räumlichen Verplanung von Veranstaltungen verletzt werden, desto schlechter ist der Stundenplan.

Die Gütekriterien sind jedoch nicht gleichwertig sondern können unterschiedlich gewichtet sein.

2.2 Constraint Logic Programming (CLP)

Das Paradigma des Constraint-Logic-Programming ist ein relativ neuer Ansatz, der das Paradigma der deklarativen logischen Programmierung um Beschränkungserfüllungstechniken erweitert. Besonders für sogenannte Constraint-Satisfaction-Probleme (CSP's) eignet sich dieser Ansatz besonders gut, da sich diese Probleme durch Einschränkungen formulieren lassen, und die Kernaufgabe einer CLP-Implementierung die Definition von Constraints über Problemvariablen ist. Im Gegensatz zu den logischen Programmiersprachen in denen nur die Unifikation (=) existiert, ist die Kernidee des CLP die implizite Definition von Relationen, bzw. Constraints über den Variablen eines Problems.

Bei einem Constraint-Satisfaction-Problem ist das Problem als Menge von Problemvariablen, über denen Einschränkungen die die Randbedingungen einer Lösung definieren definiert sind, und den über diesen Variablen definierten Constraints definiert. Beim CLP ist für jede Varibale über der Constraints definiert sind ein Wertebereich (Domain) definiert, der die Werte die die Variable annehmen kann festlegt, und kontinuierlich als auch diskret sein kann. Die Anwendung von Constraints kann diese Wertebereiche einschränken. Ist die Kardinalität aller Domains der Variablen des Problems eins, so ist diese Belegung der Variablen eine Lösung (vorausgesetzt, die Constraints definieren die Menge der Lösungen korrekt). Im Zusammenhang mit CLP wird häufig auch von CLP(X) gesprochen, wobei X die Menge ist, über der die Wertebereiche der Constraintvariablen definiert sind. CLP(R) steht z.b. für Constraint-Logic-Programmierung über der Menge der reellen Zahlen, CLP(FD) für CLP über endlichen diskreten Wertebereichen (Finite Domains).

In den allermeisten Fällen reicht das bloße Definieren des Problems als Menge von Constraintvariablen mit zugehörigen Domains und über den Variablen definierten Constraints nicht aus um eine Lösung zu finden. Es muss ein Suchprozess stattfinden bei dem die Wertebereiche der Constraintvariablen eingeschränkt bzw. den Variablen Werte aus ihrem Wertebereich zugewiesen werden. Dabei stellt ein CSP Laufzeitsystem über den sogenannten Constraint-Solver sicher, daß das System aus Variablen mit ihren aktuellen Wertebereichen und Constraints nicht inkonsistent wird, da die Einschränkung eines Wertebereiches bzw. die Zuweisung eines Wertes an eine Varaible, die zu Inkonsistezen führen wuerde fehlschlägt, und zum Backtracking führt. Dabei werden bei jeder Änderung des Wertebereiches einer Variable die zugehörigen Constraints überprüft, und evtl. Elemente aus den Wertebereichen der beteiligten Variablen entfernt, um die Konsistenz der Constraints sicherzustellen. Diese Einschränkung von Wertebereichen kann sich rekursiv fortsetzen, und wird Constraint-Propagation genannt. Bleibt hierbei eine Variable mit einem leeren Wertebereich zurück, so schlägt die ursprüngliche Einschränkung des Wertebereiches der Variable fehl. Durch die Constraint Propagation Fähigkeiten von CLP Systemem ergibt sich der Vorteil, das sich bei einer Suche Teile des Suchraums, in denen sich keine Lösung des Problems befinden kann, da das System in jedem Fall inkonsistent wäre , frühzeitig erkennen lassen.

Ein Beispiel Als Beispiel soll hier das Problem der 8 Königinnen, das eine Instanz des N-Queens-Problems mit N=8 ist, dienen, daß unter anderem auch in [AIMA95]aufgeführt ist. Dabei geht es darum 8 Königinnen auf einem Schachbrett so zu verteilen, daß keine Königin durch eine andere bedroht wird.

Die Implementierung findet mit Hilfe des auch in der Diplomarbeit verwendeten Constraint-Logic-Programming-Systems ECLiPSe statt.

Der erste Schritt zur Lösung des Problems mittels Constraint-Logic-Programming, ist die Wahl einer geeigneten Darstellung des Problems. Da in einer Zeile bzw. Spalte des Schachbretts keine zwei Königinnen gleichzeitig stehen können ohne sich gegenseitig zu bedrohen, läßt sich das Problem als Liste der Länge 8 von Constraintvariablen darstellen.Jede Variable stellt z.b. eine Spalte dar, und ihr Wert die Zeile in der die Königinn dieser Spalte positioniert ist. Dies läßt sich z.b. folgendermaßen implementieren :

```
length(Spalten,8),      %Liste von 8 Variablen
Spalten :: 1..8          %Varible

n sind Constraintvariable
```

% mit einem Wertebereich von W={1,2,3,4,5,6,7,8}

Die Aufgabe des ECLiPSe-Prädikates ::/2 besteht darin Constraintvariable mit ihren Wertebereichen zu definieren, es kann aber auch benutzt werden um den Wertebereich einer Constraintvariable zu erfragen, bzw. diesen einzuschränken.

Der nächste Schritt besteht in der Erstellung der Constraints über den Problemvariablen (hier also über den Elementen der Liste *Spalten).*Die Einschränkungen sind also :

1. Keine zwei Königinnen in einer Spalte
2. Keine zwei Königinnen in einer Reihe
3. Keine zwei Königinnen in einer Diagonale. Anders formuliert : "Ist der Abstand zweier Spalten a und b gleich A dann darf der Abstand der Reihen der Königinnen in den Spalten a und b nicht gleich A sein"

Diese Einschränkungen lassen sich z.b. folgendermassen umsetzen.

```
keineBedrohung(K1,K2,Abst) :-
K1 ## K2,              %Keine 2 Königinnen in einer Zeile
K1 - K2 ## Abst,   %Keine 2 Königinnen in einer Diagonale
K2 - K1 ## Abst.   %——— " " —————
```

Die Einschränkung Keine 2 Königinnen in einer Spalte, ergibt sich implizit aus der Umsetzung des Problems, da eine Problemvariable immer nur einen Wert aus ihrem Wertebereich annehmen kann. Das Prädikat keineBedrohung/3 muß nun noch auf alle ungeordneten Paare der Elemente der Liste *Spalten* angewendet werden, damit alle Problemvariable (wie hier notwendig) paarweise mit den o.g. Constraints belegt werden.

```
achtKoeniginnen(Spielfeld) :-
length(Spalten,8),
Spalten :: 1..8,
keineAngriffe(Spalten).

keinAngriffe([]).
keineAngriffe([Spalte|Spalten]) :-
keineAngriffe(Spalten),
```

```
keinAngriff(Spalte,Spalten,1).

keinAngriff(_,[],_).
keinAngriff(Spalte,[Spalte2|Spalten],SpaltenAbstand) :-
keineBedrohung(Spalte,Spalte2,Abstand),
NaechsterAbstand is SpaltenAbstand + 1,
keinAngiff(Spalte,Spalten,NaechsterAbstand).

% keinAngriff/3 .... wie oben definiert
```

Der letzte Schritt auf dem Weg zur Lösung ist nun das sogenannte Labeling, d.h. die Suche nach einer Lösung, bei der sukzessive die Wertebereiche der Problemvariablen eingeschränkt bzw. mit Werten aus den Wertebereichen belegt werden. Folgende Abfrage leistet dies :

```
?- achtKoeniginnen(Spielfeld),labeling(Spielfeld).
```

Das Prädikat labeling/1 ist Teil der fd-Library des ECLiPSe Systems, und führt eine einfache Backtrackingsuche im Problemraum durch, der durch die als Parameter übergebene Constraintvariablenliste dargestellt ist. Die Auswahl der jeweils den Variablen zuzuweisenden Werten (Value Selection), erfolgt sequentiell in der Reihenfolge in der die Werte in der Liste die den Wertebereich definiert, vorkommen. Labeling und Value Selection sind u.a. gute Ansatzpunkte um eigene Heuristiken zu implementieren.

2.3 Das CLP-System *ECLiPSe*

ECLiPSe steht für **ECLiPSe Common Logic Programming System** und ist ein auf der deklarativen logischen Programmiersprache PROLOG basierendes Entwicklungs und Laufzeitsystem, daß Standard PROLOG um Techniken des Constraint Logic Programming (CLP) sowie einige nicht CLP Spracherweiterungen wie z.b. Variablen mit imperativer Semantik erweitert. *ECLiPSe* wurde ursprünglich am European Computer Industry Research Centre in München als Integration verschiedener vorausgegangener Systeme zum Constraint- und Constraint-Logic-Programming (CHIP,MegaLog,SEPIA) entwickelt, und wird nun vom IC-Parc (Centre for Planning and Resource Control) am Imperial College in London gepflegt und weiterentwickelt.

ECLiPSe besteht im wesentlichen aus :

- **Kernel** D.h. Einer Edinburgh Prolog ähnlichen Prolog Implementierung die auf einem inkrementellen Compiler aufbaut der den *ECLiPSe* Code in Maschinencode einer virtuellen Maschine (WAM) übersetzt.
- **Libraries** Für die verschiedenen von *ECLiPSe* unterstützten CLP(X) Instanzen wie z.b. FD (Constraints über endlichen Wertebereichen), R (Constraints über Reellen Zahlen) etc. . Diese Libraries enthalten neben den entsprechenden Constraint-Prädikaten auch die Constraint-Solver, also die Algorithmen mit denen Constraints überwacht und propagiert werden.
- **Libraries** Die die Entwicklung eigener Constraint-Solver und die Entwicklung komplexer disjunktiver Constraints unterstützen.
- **Libraries** Die verschiedene Such und Optimierungsalgorithmen zur Verfügung stellen, bzw. die Entwicklung eigener Such- und Optimierungsalgorithmen unterstützen.
- **Libraries** Zur Unterstützung des HHTP Protokols, für erweiterte IO-Funktionalität (Sockets, Streams etc.), ...
- **Entwicklungsumgebungen** Eine Konsolen- sowie eine GUI-orientierte Benutzerschnittstelle stehen zur Verfügung, die auch einen auf dem bekannten Box-Model basierten Debugger und einige weitere Tools (Prädikatenbrowser, Inspector für Prolog-Terme, Profiler etc.) zur Verfügung stellen.

ECLiPSe ist ein hybrides System, da es ein Constraint-Logic-Programming System ist, mit dem es möglich ist Constraints über verschiedenen Problemklassen gleichzeitig zu verwenden. Also z.b. in einem System Constraints über endlichen Wertebereichen, Mengen und Reellen Zahlen.

ECLiPSe unterstützt die Verwendung von Constraints und Repair-Constraints. Constraint-Solver für Repair-Constraints lassen im Gegensatz zu den Solvern für Constraints auch Inkonsistenzen innerhalb des Systems aus Constraintvariablen und Constraints zu, und verwalten sogenannte Conflict-Sets, in denen die Constraints gesammelt werden die an Konflikten beteiligt sind. Mit Hilfe von Repair-Constraints lassen sich z.b. auch lokale Suchverfahren wie z.b. Simulated Annealing Implementieren.

Neben den Constraint-Logic-Programming Erweiterungen erweitert *ECLiPSe* PROLOG auch um einige CLP unabhängige Spracheigenschaften

- **Unterstützung der parallelen Ausführung von Code**. ECLiPSe unterstützt Oder-Parallelität, d.h. der Nichtdeterminismus von Prolog kann anstatt über Backtracking auch über das parallele Ausführen der unterschiedlichen Klauseln eines Prädikates erreicht werden.

- **Event- und Interruptsystem** Interrupts sind asynchrone Unterbrechungen des Programmflusses z.b. durch Betriebssystemereignisse, die durch einen Interrupthandler behandelt werden der asynchron in einer eigenen ECLiPSe Umgebung gestartet wird. Events können synchron vom Programm erzeugt werden. Die Eventhandler werden synchron in derselben ECLiPSe Umgebung gestartet, in der auch der Event erzeugt wurde.

- **Schleifenkonstrukte** Sollen die sonst zur iteration benötigten Hilfsprädikate weitgehend überflüssig machen, und werden intern in rekursive Hilfsprädikate übersetzt.

- **Soft Cut** Implementiert durch das Prädikat *->/3, daß sich wie ->/3 verhält, nur daß im Falle von *goalA -> goalB;goalC* ein Backtracking weder aus goalB noch aus goalC zurück nach A erfolgen kann und im Falle des SoftCut *goalA *-> goalB;goalC* ein Backtracking von goalB zurück nach goalA nicht aber von goalC nach goalA erfolgen kann.

- Modulsystem, Arrays, Strings, globale nicht deklarative Variable, Konstrukte zum einfachen Zugriff auf Prolog-Terme.

3 Anforderungsanalyse

Bei der Anforderungsanalyse geht es darum das dem zu erstellenden Softwaresystem zugrundeliegede Problem zu erfassen, darzustellen und die vom System zu erfüllenden Eigenschaften zu definieren. Die Analyse dient dann als Basis für den folgenden Schritt, also das den Entwurf des Systems. Die Anforderungsanalyse soll jedoch keine technischen Details bzw. Aussagen über die spätere softwaretechnische Struktur des zu erstellenden Systems machen. Dies ist den nachfolgenden Phasen vorbehalten. Im Nachfolgenden werden das Problem und die Anforderungen der Stundenplanung detailiert dargestellt, und zum Abschluss dieses Abschnittes die Anforderungen zusammengefaßt.

3.1 Veranstaltungen und Hörergruppen

Die Studierenden an der Fachhochschule/PTL Wedel lassen sich nach Fachbereich, Fachsemester und evtl. nach der Vertiefungsrichtung im Hauptstudium in Gruppen einteilen. Momentan gibt es an der Fachhochschule die Studienfächer (Kürzel in Klammern) Wirtschaftsinformatik (wi), Technische Informatik (ii), Medieninformatik (mi), Physikalische Technik (pi), Wirtschaftsingenieurwesen (di), sowie die Aufbaustudiengänge Wirtschaftsingenenieurwesen (ai) und Umwelttechnik (ui). Die Fachhochschule Wedel steht in engem räumlichen und personellen Zusammenhang mit der Physikalisch Technischen Lehranstalt (PTL) Wedel. Die Schüler an der PTL sind Informatik Assistenten (ia), Physikalisch Technische Assistenten (pa) sowie Technische Assistenten für Elektronik und Datentechnik (ea). Dozenten die an der FH-Wedel Veranstaltungen halten halten u.U. auch Veranstaltungen an der PTL und umgekehrt. In einigen Fällen werden Veranstaltungen von Studenten der FH und Schülern der PTL gleichermassen besucht. Der zu erstellende Stundenplan ist daher ein integrierter Stundenplan für die FH-Wedel/PTL.

Die zu haltenden Veranstaltungen eines Semesters ergeben sich aus dem Studienplan der FH bzw. dem Lehrplan der PTL, der für jedes Fachsemester einer Fachrichtung die in diesem Semester vorgesehenen Vorlesungen zuordnet. Grundsätzlich können alle Veranstaltungen entweder zwingend im Studienplan vorgeschrieben oder freiwillig sein. Für zwingend vorgeschriebene Veranstaltungen muss sichergestellt sein, das alle Studenten für die diese Veranstaltungen vorgesehen sind, diese auch hören bzw. an ihr teilnehmen können. D.h. Die Situation das Studenten an mehreren für sie vorgesehenen Vorlesungen gleichzeitig teilnehmen müssten, ist für im Studienplan zwingend vorgesehene Vorlesugen auf jeden Fall zu vermeiden. Für freiwillige Veranstaltungen ist diese Einschränkung nur nach Möglichkeit, nicht aber zwingend einzuhalten. Da die Veranstaltungen des Studienplanes bis auf wenige Ausnahmen zwingend vorgeschrieben sind wird in dieser Diplomarbeit jedoch davon ausgegangen, daß alle Veranstaltungen zwingend vorgesehen sind. Die Auswertung des Studienplanes bzw. die überprüfung der Konsistenz der zu haltenden Vorlesungen mit dem Studienplan ist jedoch keine Anforderung an das zu erstellende System.

Die an der FH-Wedel/PTL im Stundenplan zu verteilenden Veranstaltungen lassen sich einteilen in

1. Vorlesungen. Sind Lehrveranstaltungen die von einem Dozenten gehal-

ten werden, und gewisse Mindestanforderungen an die Räume stellen (Zur Verfügung stehender Tafelplatz, Overhead)

2. Seminare. Sind Lehrveranstaltungen, die von einem Dozenten betreut, aber von den Teilnehmern des Seminars gehalten werden.

3. Praktika. Werden von einem oder mehreren Assistenten betreut, und stellen meist spezielle Anforderungen an die Ausstattung der Räume in denen sie stattfinden können (Rechenzentren,Labore, etc.)

Da diese Unterscheidung jedoch bis auf die unterschiedlichen Ressourcenanforderungen auf den zu erstellenden Stundenplan keinen Einfluß hat, muß das Stundenplanungssystem keine explizite Unterscheidung von Veranstaltungen nach Seminaren,Vorlesungen und Praktika unterstützen.

An einer Veranstaltung nehmen also ein oder mehrere Assistenten und/oder Dozenten und eine Menge von Hörergruppen teil. Eine Hörergruppe könnte z.b. folgendermassen aussehen $\{ii1, wi1, mi2, pi3\}$ d.h. an einer Veranstaltung mit dieser Hörergruppe würden angehende Technische und Wirtschaftsinformatiker im 1. Semester, Medieninformatiker im 2. sowie Phyisikingenieure im 3. Semester teilnehmen.

Weiterhin kann es vorkommen daß für eine Vorlesung kein fester Termin in einem festen Raum notwendig ist, da die Teilnehmer der Vorlesung dies individuell vereinbaren.

Veranstaltungen haben einen Namen und Anforderungen bezüglich der für die Veranstaltung zur Verfügung stehenden Resourcen.

- Teilnehmer der Veranstaltung (Dozenten/Assistenten/Studierende) müssen der Veranstaltung zur Verfügung stehen, d.h. nicht freiwillige Veranstaltungen können auf keine Ressource (Hörergruppe) aus der Gruppe der Studierenden verzichten. D.h. diese Ressourcen müssen zum Zeitpunkt der Veranstaltung zeitlich verfügbar sein.

- Räumliche Ressourcen. Die maximale Teilnehmerzahl, die sich aus der Summe aller Grössen aller Teilnehmergruppen ergibt, muss im Veranstaltungsraum Platz finden. Andere benötigte Ressourcen der Veranstaltung (wie z.b. Videoprojektoren mit hoher Auflösung o.ä.) müssen im Veranstaltungsraum verfügbar sein.

- Eine Veranstaltung kann aus einem oder mehreren Blöcken bestehen, wobei ein Block eine zeitliche und räumliche Einheit bildet, d.h sich über eine oder mehrere zusammenhängende Zeiteinheiten erstreckt und über diese Zeit in einem Raum stattfindet. Es kann erlaubt oder auch verboten sein, daß mehrere Blöcke ein und derselben Vorlesung an einem Tag stattfinden. Die Aufteilung der Gesamtzahl der Vorlesungseinheiten einer Vorlesung pro Woche steht vor der Erstellung eines Stundenplanes fest. Beispiel: Vorlesung X hat pro Woche 3 Zeiteinheiten und besteht aus einem 2er und einem 1er Block die nicht an einem Tag gemeinsam stattfinden dürfen.

Eine Veranstaltung wird also durch folgende Parameter definiert.

- Eindeutiger Name der Veranstaltung
- Menge der Dozenten/Assistenten die die Veranstaltung halten
- Menge der an der Veranstaltung teilnehmenden Gruppen
- Ressourcenanforderungen. Zwingend benötigte Ressourcen der Veranstaltung bezüglich des Veranstaltungsraumes

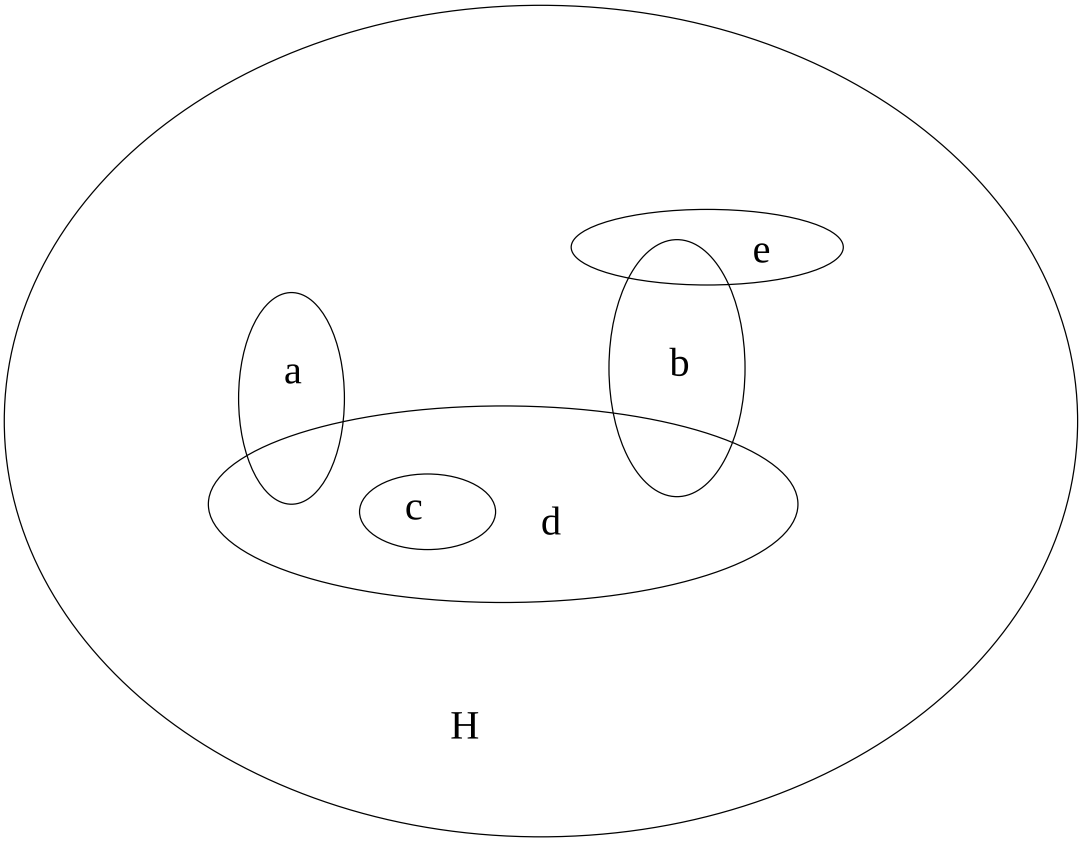
Neben den zu haltenden Veranstaltungen müssen auch die Hörergruppen und Dozenten frei definierbar sein, da sich das Ausbildungsangebot bzw. die Studiendauer von Fachrichtungen sowie die Menge der Dozenten/Assistenten ändern können. Eine Unterscheidung zwischen Dozenten und Assistenten ist jedoch für das Stundenplanungssystem nicht wichtig, da es für den Stundenplan ohne Unterschied ist, ob eine Veranstaltung von einem Dozenten oder einem Assistenten gehalten wird.

Dozenten und Assistenten sind durch ihr Namenskürzel charakterisiert, während Hörergruppen durch ihr Gruppenkürzel aus dem Studienrichtung und Fachsemester hervorgehen (z.b. ii6 für Technische Informatik 6. Semester) sowie die Anzahl an Hörern dieser Gruppe charakterisiert werden.

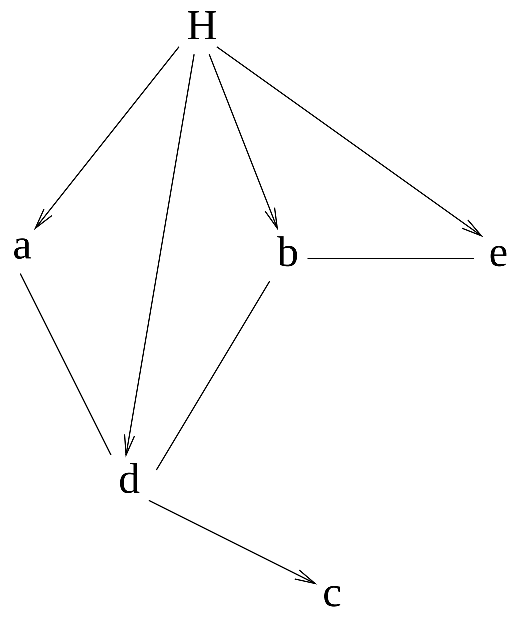
Einteilung von Gruppen in Unter- bzw. Subgruppen Es kann notwendig sein Gruppen in Unter- bzw. Subgruppen aufzuteilen. So besteht z.b. im Hauptstudium der Fachrichtung Technische Informatik die Möglichkeit den Studienschwerpunkt auf Hardware oder Software zu legen. Je nach Wahl des Studienschwerpunktes gibt es dann Veranstaltungen im Studienplan die nur für Studenten jeweils eines Studienschwerpunktes vorgesehen sind. Zwei solche Veranstaltungen sind z.b. das Softwarepraktikum (u.a. nehmen hier Technische Informatiker des Studienschwerpunktes Software teil) sowie das Hardwaredesign-Praktikum, an dem Technische Informatiker des Studienschwerpunktes Hardware teilnehmen. Würde nun die Gruppe *ii6* (Technische Informatik 6. Semester) für beide o.g. Veranstaltungen vorgesehen werden, da ja beide Studienschwerpunkte zu dieser Gruppe gehören, könnten diese Veranstaltungen nicht parallel stattfinden, obwohl dies im Stundenplan möglich sein sollte.

Eine Möglichkeit dieses Problem zu Lösen ist die Unterteilung von Gruppen in Subgruppen, die dann an parallel stattfindenden, bzw. sich überschneidenden Veranstaltungen teilnehmen können. Für obiges Beispiel könnte man also zum Beispiel die Gruppe *ii6* in die Untergruppen *ii6Hardware* sowie *ii6Software* unterteilen, so daß Veranstaltungen mit diesen beiden Gruppen parallel zueinander, nicht jedoch parallel zu Veranstaltungen mit der Obergruppe *ii6* stattfinden könnten.

Bei der Bildung von Subgruppen kann jedoch das Problem auftreten das Subgruppen die aus verschiedenen Gründen von einer Obergruppe gebildet wurden untereinander nicht disjunkt sind. Es könnte z.b. sein, daß zwei weitere *ii6* Subgruppen *ii6Seminar_A* sowie *ii6Seminar_B* existieren, um Gruppen für zwei Seminare zu definieren, aus denen Studenten der Gruppe ii6 eines auswählen müssen. Es kann dann vorkommen das die Gruppen ii6Hardware sowie ii6Software zu den Gruppen ii6Seminar_A sowie ii6Seminar_B nicht disjunkt sind, weil an beiden Seminaren die Studienrichtungen Hardware und Software teilnehmen könnten. Wenn dann auch noch Subruppen als Gruppen die wiederum in Subgruppen unterteilt werden können angesehen werden, könnten sich beispielsweise Situationen wie im folgenden Mengen-Diagramm dargestellt ergeben :



H ist dabei die Hauptgruppe alles andere sind Subgruppen von H. Stellt man diese Situation als Graph dar, in dem Knoten Gruppen bzw. Subgruppen sind , gerichtete Kanten Ober-Untergruppenrelationen darstellen, und ungerichtete Kanten nicht disjunkte Gruppen verbinden die in keiner Ober-Untergruppenrelation zueinander stehen, ergibt sich folgender Graph:

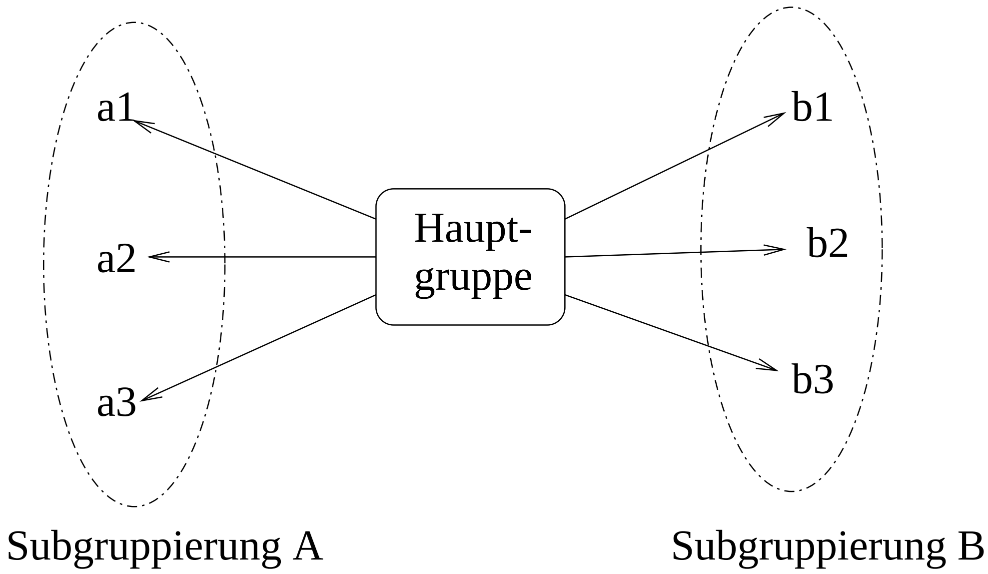


Vorlesungen die zwei Gruppen G1 und G2 gemeinsam haben dürfen nicht stattfinden wenn im zugehörigen Gruppengraphen ein Pfad aus ausschließlich gerichteten Kanten von G1 nach G2 oder/und von G2 nach G1 existiert, oder G1 und G2 durch eine ungerichtete Kante verbunden sind. In der Praxis läßt sich dieser doch recht komplexe Ansatz jedoch wesentlich vereinfachen, da meist ohnehin nicht bekannt ist welche Subgruppen nicht disjunkt sind, und somit die Konstruktion eines Graphen, wie oben dargestellt, unmöglich ist.

Vereinfachtes Subgruppenmodell Aus diesem Grund wird von folgenden Voraussetzungen ausgegangen :

1. Subgruppen von Subgruppen sind nicht möglich
2. Subgruppen werden jeweils in einem mit einem Namen versehenen Kontext gebildet, und sind innerhalb dieses Kontext zueinander disjunkt.
3. Subgruppen einer Obergruppe aber aus verschiedenen Kontexten sind grundsätzlich nicht disjunkt zueinander.

Für zwei verschiedene Kontexte mit jeweils drei Untergruppen einer gemeinsamen Obergruppe ergibt sich folgende Darstellung:



Die Kontexte haben in der Darstellung die Namen *SubgruppierungA* sowie *SubgruppierungB* .

Bezogen auf das weiter oben angeführte Beispiel lassen sich nun zum Beispiel zwei Kontexte II_6_HardSoft sowie II_6_Seminar bilden, wobei die Subgruppen ii6Hardware sowie ii6Software zu II_6_HardSoft und ii6Seminar_A und ii6Seminar_B zu II_6_Seminar gehören. Werden diese Gruppen als Teilnehmer für die Vorlesungen Softwarepraktikum (ii6Software), Hardwaredesing-Praktikum (ii6Hardware), Seminar_A (ii6Seminar_A) sowie Seminar_B (ii6Seminar_B) verwendet, können sowohl Seminare parallel zueinander als auch die Praktika parallel zueinander, nicht jedoch die Seminare und Praktika sowie die restlichen Vorlesungen für ii6 parallel zueinander stattfinden.

3.2 Räume, Vorlesungstage und Zeiteinheiten

Räume lassen sich einteilen in

- Hörsäle
- Rechenzentren
- Labore
- Seminarräume

Diese Einteilung ist jedoch für den Stundenplan von untergeordneter Bedeutung, da eine Entscheidung ob eine Veranstaltung in einem Raum stattfinden kann an den Ressourcenanforderungen der Vorlesung und den Ressourcenangeboten von Räumen ausgerichtet werden kann.

Die Parameter die einen Raum definieren sind also

- Größe des Raumes. D.h für Hörsäle und Seminarräume die Gesamtzahl an Hörern die in dem Raum Platz finden, und für Rechenzentren und Labore die Anzahl an Arbeitsplätzen.
- Ressourcenangebot des Raumes. Menge von Ausrüstungsgegenständen und speziellen Eigenschaften des Raumes z.b.: {Overhead, Netzzugang, Klimatisiert} o.ä.
- Eindeutiger Name des Raumes.

Räume sollen frei definierbar sein, da sich Anzahl, Art und Ausstattng der Räume ändern können.

Vorlesungstage sind duch ihren Namen definiert, und es wird davon ausgegangen, daß jeder Tag bezüglich der Zeiteinteilung in Zeiteinheiten gleich dem anderen ist. Zeiteinheiten sind wie Tage ebenfalls durch ihren Namen definiert, da aber Vorlesungsblöcke sich über mehrere Zeiteinheiten eines Tages nicht jedoch über mehrere Tage erstrecken können ist die Ordnung von Zeiteinheiten relevant. Bestimmte Zeiteinheiten sind gegenüber anderen Zeiteinheiten zu präferieren. Die Vorlesungseinheit die die Mittagspause darstellt, und Vorlesungseinheiten am Spätnachmittag sind z.b mit möglichst wenigen Vorlesungseinheiten zu belegen. Die Arbeitstage Montag bis Freitag sind dem Samstag vorzuziehen. Eine Vorlesung sollte i.a. jedoch besser zur Mittagszeit als am Samstag stattfinden. Die Zeiteinheiten und Wochentage sind also gewichtet.

3.3 Allgemeine und spezifische Einschränkungen

3.3.1 Allgemeines zu Einschränkungen

Für alle laut Vorlesungsplan nicht freiwilligen Veranstaltungen ist sicherzustellen, daß die für die Veranstaltungen vorgesehenen Hörergruppen an allen für sie vorgesehenen Veranstaltungen teilnehmen können. D.h. daß sich Veranstaltungen die gemeinsame Hörergruppen haben nocht gleichzeitig stattfinden bzw. sich nicht überschneiden dürfen. Zwei Veranstaltungen haben dann gemeinsame Hörergruppen wenn die Mengen ihrer Hörer nicht disjunkt sind.

Die allgemeinen Einschränkungen, deren Nichteinhaltung zu einem nicht durchführbaren Stundenplan führt sind also folgende :

1. Nicht freiwillige Vorlesungen mit nicht disjunkten Hörergruppen oder nicht disjunkten Assistenten/Dozentenmengen dürfen nicht gleichzeitig stattfinden.

2. In einem Raum darf nicht mehr als eine Veranstaltung gleichzeitig stattfinden.

3. Veranstaltungen finden nur in Räumen statt, die die Ressourcenanforderungen der Veranstaltung erfüllen.

4. Besteht eine Vorlesung aus mehreren Blöcken, so dürfen sich die Blöcke einer Vorlesung nicht zeitlich überlappen.

5. Der Platzbedarf von Veranstaltungen ist kleiner oder gleich dem Platzangebot des Raumes in dem die Veranstaltung stattfindet.

6. Alle Hörergruppen müssen die Möglichkeit haben in der Mensa zu Mittag zu essen. D.h. in der Mittagszeit (Vorlesungen mit Beginn um 11:00-13:45) muss jede Hörergruppe mindestens eine freie Vorlesungseinheit haben.

Spezifische Einschränkungen sind solche, die für einen konkret zu erstellenden Stundenplan individuell definierbar, und nicht etwa allgemeingültig sind. Spezifische Einschränkungen lassen sich einteilen in

1. **Einschränkungen deren Nichteinhaltung zu einem nicht durchführbaren Stundenplan führt**. Eine solche Einschränkung könnte z.b. die Tatsache sein, daß ein externer Dozent definitiv nur zu einem bestimmten Termin eine Vorlesung halten kann.

2. **Einschränkungen deren Nichteinhaltung zu einem qualitativ schlechterem Stundenplan führen**, die jedoch nicht zwingend erfüllt sein müssen. Eine solche Einschränkung könnte z.b. der Wunsch eines Dozenten sein eine bestimmte Vorlesung nur Vormittags zu halten.

Einschränkungen der zweiten Art können gewichtet sein, daß heißt die Qualitätsverschlechterung des Stundenplanes bei nichteinhaltung solcher Einschränkungen ist nicht für alle Einschränkungen gleich groß.

Weiterhin lassen sich spezifische Einschränkungen einteilen in

1. **Einschränkungen bezüglich der zeitlichen Verfügbarkeit von Dozenten bzw. Assistenten (also allgemein Mitarbeitern)**. Eine solche Einschränkung könnte z.b. sein, daß ein Dozent an bestimmten Tagen zu bestimmten Zeiten keine Vorlesungen halten kann oder möchte.

2. **Einschränkungen bezüglich der zeitlichen Verplanung von Vorlesungen**. Ein Beispiel für eine solche Einschränkung könnte z.b. sein, daß eine bestimmte Vorlesung nur Vormittags stattfinden kann oder soll.

3.3.2 Die benötigten spezifischen Einschränkungen

Die Datenerfassung für zu erstellende Stundenpläne findet über den Abgleich von Erfassungsbögen die von Dozenten/Assistenten ausgefüllt werden, mit dem Studienplan statt. Mit Hilfe der Erfassungsbögen definieren die Dozenten/Assistenten die zu haltenden Vorlesungen, deren Ressourcenanforderungen und die spezifischen Einschränkungen bezüglich ihrer Person und der zeitlichen Verplanung der definierten Veranstaltungen. Um die benötigten spezifischen Einschränkungen bezüglich der Dozenten/Assistenten bzw. der Veranstaltungen zu erhalten, wurde ein realer Satz der o.g. Erfassungsbögen analysiert. Die Analyse ergab, daß folgende spezifische Einschränkungen vom Stundenplanungssystem zur Verfügung zu stellen sind, bzw. benötigt werden.

Vorlesung_Um Mit Hilfe dieser Einschränkung kann eine Vorlesung auf eine bestimmte Zeiteinheit beschränkt werden. Sollen mehrere optionale Zeiteinheiten definiert werden, so geschieht dies nicht mit dieser Einschränkung, sondern mit Hilfe der Einschränkung Vorlesung_Nicht_Um. Die Einschränkung ist spezifiziert durch den Namen der Vorlesung und die Zeiteinheit auf die diese Vorlesung gelegt werden soll. Für Vorlesungen aus mehr als einem Block gilt die Einschränkung für alle Blöcke der Vorlesung.

Vorlesung_Am Mit Hilfe dieser Einschränkung kann eine Vorlesung auf einen bestimmten Vorlesungstag beschränkt werden. Sollen mehrere optionale Vorlesungstage definiert werden, so geschieht dies nicht mit dieser Einschränkung, sondern mit Hilfe der Einschränkung Vorlesung_Nicht_Am. Die Einschränkung ist spezifiziert durch den Namen der Vorlesung und denVorlesungstag auf den diese Vorlesung gelegt werden soll. Für Vorlesungen aus mehr als einem Block gilt die Einschränkung für alle Blöcke der Vorlesung.

Vorlesung_Am_Um Mit dieser Einschränkung lassen sich Vorlesungstag sowie die Zeiteinheit auf die die Vorlesung gelegt wird vorbelegen. Die Zeiteinheit bezieht sich auf die Anfangszeit der Vorlesung. Mehrere optionale Termine werden nicht mit diesem, sondern mit Hilfe der Einschränkung Vorlesung_Nicht_Am_Um definiert. Diese Einschränkung ist nur anwendbar auf Vorlesungen die aus nicht mehr als einem Block bestehen, da sonst die Blöcke einer Vorlesung parallel stattfinden würden, was sinnlos ist. Die explizite Angabe von Terminen für einzelne Blöcke von Vorlesungen ist nicht möglich.

Vorlesung_Nicht_Um Diese Einschränkung bezieht sich auf eine Vorlesung und eine Menge von Zeiteinheiten, zu denen die Vorlesung nicht stattfinden darf bzw. soll. D.h. Es darf kein Block einer Vorlesung, auch nicht teilweise,in diese Zeiteinheiten fallen. Auch für die Definition von Zeitintervallen in die hinein eine Vorlesung gelegt werden soll bzw. darf läßt sich diese Einschränkung verwenden, indem die ausserhalb dieser Intervalle liegenden Vorlesungseinheiten mit Hilfe dieser Einschränkung ausgeschlossen werden.

Vorlesung_Nicht_Am Diese Einschränkung bezieht sich auf eine Vorlesung und eine Menge von Vorlesungstagen,an denen die Vorlesung nicht stattfinden darf bzw. soll. Auch für die optionale Definition mehrerer Tage an denen die Vorlesung stattfinden kann läßt sich diese Einschränkung verwenden, indem die anderen Vorlesungstage mit Hilfe dieser Einschränkung ausgeschlossen werden.

Vorlesung_Nicht_Am_Um Mittels dieser Einschränkung lassen sich bestimmte Termine ausschliessen auf die kein Block der Veranstaltung, für die diese Einschränkung gilt fallen darf. Diese Einschränkung bezieht sich immer auf eine Vorlesung, einen Tag aber möglicherweise mehrere Zeiteinheiten, so daß es durch mehrere solcher Einschränkungen für eine Vorlesung möglich ist an verschiedenen Tagen unterschiedliche Zeiteinheiten für die Vorlesung auszuschliessen. Falls eine Menge von optionalen Terminen für die Vorlesung festgelegt werden soll, so kann auch dies mit Hilfe dieser Einschränkung definiert werden, indem die ausserhalb dieser Menge liegenden Termine mit Hilfe dieser Einschränkung für die Vorlesung ausgeschlossen werden.

Vorlesungen_Hintereinander Diese Einschränkungen bezieht sich auf zwei Vorlesungen, und legt fest daß diese Vorlesungen in der Reihenfolge ihrer Angabe in dieser Einschränkung und zeitlich auber nicht räumlich zusammenhängend im Stundenplan auftauchen sollen. Wie auch bei der Einschränkung Vorlesung_Am_Um ist diese Einschränkung nur auf Vorlesungen anwendbar die aus nicht mehr als einem Block bestehen.

Vorlesungen_Nicht_Zusammenhängend Bezieht sich auf zwei Vorlesungen die im Gegensatz zur Einschränkung Vorlesungen_Hintereinander auch aus mehr als einem Block bestehen können, und legt fest, daß ein Block der einen Vorlesung zu keinem Block der anderen Vorlesung in direktem zeitlichen Zusammenhang stehen darf. Beispiel : Das Ende eines Blockes a der Vorlesung A darf also weder mit dem Anfang eines Blockes b der Vorlesung zusammenfallen, noch darf das Ende von Block b mit dem Anfang von Block a zusammenfallen.

Vorlesungen_Am_Selben_Tag Bezieht sich auf zwei Vorlesungen, und legt fest das diese beiden Vorlesungen gemeinsam an einem Tag stattzufinden haben, bzw. stattfinden sollen. Diese Einschränkung beszieht sich auf alle Blöcke der beiden Vorlesungen, d.h. alle Blöcke der beiden Vorlesungen finden an einem Tag statt.

Vorlesungen_Nicht_Am_Selben_Tag Diese Einschränkung bezieht sich auf zwei Vorlesungen, die jeweils auch aus mehr als einem Block bestehen können, und legt fest, daß an keinem Tag Blöcke aus beiden Vorlesungen stattfinden dürfen, bzw. sollen.

Dozent_Nicht_Am Diese Einschränkung bezieht sich auf einen Dozenten bzw. Assistenten und eine Menge von Tagen, an denen der Dozent/Assistent keine Veranstaltungen halten kann bzw. soll. Dies schränkt natürlich auch die Veranstaltungstage aller Veranstaltungen an denen der betreffende Dozent/Assistent beteiligt ist auf die Tage ein, die in der Menge von Tagen auf die sich diese Einschränkung bezieht nicht auftauchen. Soll eine Menge von Tagen definiert werden an denen ein Dozent/Assistent zur Verfügung steht, so läßt sich auch dies mit Hilfe dieser Einschränkung realisieren, indem die Tage die nicht Teil dieser Menge sind mit Hilfe dier Einschränkung für den Dozenten/Assistenten ausgeschlossen werden.

Dozent_Nicht_Um Diese Einschränkung bezieht sich auf einen Dozenten bzw. Assistenten und eine Menge von Zeiteinheiten, an denen der Dozent/Assistent keine Veranstaltungen halten kann bzw. soll. Dies schränkt natürlich auch die Zeiten ein in auf die Blöcke von Veranstaltungen an denen der Dozent/Assistent teilnimmt fallen dürfen. Sollen Zeitintervalle definiert werden während denen der Dozent/Assistent zur Verfügung steht, so läßt sich auch dies mit Hilfe dieser Einschränkung realisieren, indem die Zeiten die ausserhalb dieser Intervalle liegen mit Hilfe dier Einschränkung für den Dozenten/Assistenten ausgeschlossen werden.

Dozent_Nicht_Am_Um Diese Einschränkung bezieht sich auf einen Dozenten/Assistenten, einen Tag und eine Menge von Zeiteinheiten, an denen der Dozent/Assistent keine Veranstaltungen halten kann bzw. soll. Die Menge der Zeiteinheiten wird jeweils für den angegebenen Tag für den Dozenten/Assistenen ausgeschlossen, so daß es möglich ist über mehrere Dozent_Nicht_Am_Um Einschränkungen an verschiedenen Tagen unterschiedliche Zeiten für den Dozenten/Assistenten auszuschliessen. Der Ausschluss der so definierten Termine bezieht sich natürlich auch auf die Blöcke aller Vorlesungen an denen der Dozent/Assistent teilnimmt.

Mindestens_N_FreieTage Diese Eischränkung bezieht sich auf einen Dozenten/Assistenten sowie eine positive ganze Zahl N von Vorlesungstagen, und legt fest, daß ein Dozent mindestens N freie Tage *zusätzlich* zu den mit der Einschränkung Dozent_Nicht_Am definierten freien Tagen pro Woche hat.

Vorlesungsstunden_Vor_Vorlesung_Freihalten Es ist möglich, daß ein Dozent/Assistent vor einer bestimmten Veranstaltung Vorbereitungszeit benötigt. Diese Einschränkung bezieht sich aus diesem Grund auf einen Dozenten und eine Vorlesung, und legt fest daß der betreffende Dozent direkt vor der Vorlesung an keiner Veranstaltung teilnimmt.

Alle Einschränkungen beziehen sich immer auf die komplette Vorlesung, also auf alle Blöcke aus denen sie besteht.

Einschränkungen können, solange sie in keinem logischen Konflikt stehen, miteinander kombiniert werden. Soll z.b. definiert werden, daß eine Vorlesung A grundsätzlich nicht Nachmittags, nicht am Dienstag und am Mittwoch nicht um 8:00 stattfinden darf, so sind die Einschränkungen Vorlesung_Nicht_Um (Um Nachmittagsvorlesungen auszuschliessen), Vorlesung_Nicht_Am (Um den Dienstag auszuschliessen) sowie Vorlesung_Nicht_Am_Um (Um die Vorlesungseinheit Mittwoch um 8:00 auszuschliessen) für die Vorlesung A zu kombinieren. d.h. es sind mehrere Einschränkungen für Vorlesung A zu definieren.

Es ist möglich, daß in der Zukunft weitere Einschränkungen benötigt werden. Eine Erweiterung des Systems um weitere verarbeitbare spezifische und/oder allgemeine Einschränkungen sollte möglich sein.

3.4 Zusammenfassung der Anforderungen

Im Folgenden werden die Anforderungen an das zu erstellende Stundenplanungssystem, unter Berücksichtigung der erwähnten Vereinfachungen, zusammengefasst.

1. Hörergruppen müssen zusammen mit ihrer Größe frei definierbar sein.

2. Die Definition von Subgruppen muss, wie unter VEREINFACHTES SUBGRUPPENMODELL auf Seite 27 beschrieben, möglich sein.

3. Dozenten und Assistenten müssen frei definierbar sein, wobei keine Unterscheidung zwischen Dozenten und Assistenten durch das Stundenplanungssystem notwendig ist.

4. Räume müssen, zusammen mit den durch den jeweiligen Raum zur Verfügung gestellten Ressourcen, frei definierbar sein.

5. Die Vorlesungstage müssen unter Angabe ihres Namens frei definierbar sein.

6. Die Zeiteinheiten eines Tages müssen unter Angabe ihrer Namen frei definierbar sein. Aus der Definition der Zeiteinheiten muss ihre Reihenfolge hervorgehen, da die zeitliche Abfolge von Zeiteinheiten für die Stundenplanerstellung relevant ist. Es wird davon ausgegangen, daß für jeden der definierten Tage die Einteilung in Zeiteinheiten identisch ist.

7. Für Räume, Zeiteinheiten sowie Vorlesungstage müssen Gewichte definiert werden können, so daß das Stundenplanungssystem die Kosten der Verteilung von Veranstaltungen auf Tage,Zeiten und Räume berücksichtigen kann.

8. Im Stundenplan zu verplanende Vorlesungen müssen unter Angabe ihres Namens, der teilnehmenden Hörergruppen/Dozenten/Assistenten, der Ressourcenanforderugen der Vorlesung, sowie ihrer Aufteilung in Blöcke, frei definierbar sein.

9. Sollte es dem Stundenplanungssystem nicht möglich sein einen vollständigen Stundenplan zu erstellen, so ist zumindest der bis zur Aufgabe des Stundenplanungsprozesses erstellte Partielle Stundenplan unter Angabe der nicht verplanten Veranstaltungen auszugeben.

10. Sollte es Aufgrund von Konflikten in der Einschränkung des Stundenplanes nicht möglich sein einen Stundenplan zu erstellen, so sollte dieser Konflikt gemeldet werden.

11. Die unter ALLGEMEINES ZU EINSCHRÄNKUNGEN auf Seite 29 aufgeführten Einschränkungen sind vom System zu berücksichtigen.

12. Die unter DIE BENÖTIGTEN SPEZIFISCHEN EINSCHRÄNKUNGEN 31 auf Seite aufgeführten spezifischen Einschränkungen müssen vom System verarbeitet werden können. D.h. diese Einschränkungen müssen für zu erstellende Stundenpläne definierbar sein.

13. Die oben genannten Definitionen in Bezug auf einen zu erstellenden Stundenplan, müssen getrennt vom eigentlichen Stundenplanungssystem definierbar sein.

14. Die Ausgabe des Ergebnisses sollte in Übersichtlicher Form z.b. im HTML-Format als Tabelle möglich sein.

Vereinfachungen

- Keine Unterscheidung von Veranstaltungen nach Vorlesungen, Seminaren und Praktikas
- Keine Unterscheidung von Assistenten und Dozenten
- Nur harte Einschränkungen. D.h. Eine Gewichtung von spezifischen Anforderungen ist nicht geplant. Das System sollte jedoch so entworfen

- werden, daß eine Erweiterung des Stundenplanungssystems in Richtung der Einführung von Gewichten für spezifische Anforderungen, ohne erheblichen Aufwand möglich ist.
- Keine Unterscheidung von freiwilligen und nicht freiwilligen Veranstaltungen. Die Erweiterung des Systems um die Unterscheidung solcher Vorlesungen sollte jedoch leicht möglich sein.

4 Entwurf

In diesem Abschnitt wird der Entwurf des Stundenplansystems dargestellt. Ziel des Entwurfes ist es, für die während der Analysephase definierten Anforderungen, das technische Konzept zur Realisierung dieser Anforderungen zu erstellen, welches dann während der Implementierung in realen Programmcode umgesetzt wird.

4.1 Grundsätzlicher Ablauf von CLP-Programmen

Grundsätzlich läßt sich der Ablauf von CLP-Programmen in vier Phasen einteilen. Dies sind Aufbau der Constraintvariablen, Erstellung von Constraints über diesen Variablen, Sukzessive belegung der Constraintvariablen mit Werten aus ihrem Wertebereich (Lösungssusche), wobei die Constraint-Prooagation-Fähigkeiten des CLP Systems zum Einsatz kommen, und schliesslich die Ausgabe des Ergebnisses. Im Folgenden sollen wesentliche Entwurfentscheidungen wie die Wahl der Problemrepräsentation, die Darstellung der Constraints etc. dargestellt werden. Die Aufteilung der Darstellung dieser Entwurfentscheidungen orientiert sich dabei an den vier oben genannten Phasen des Ablaufes eines CLP-Programmes.

4.1.1 Aufbau von Constraintvariablen mit ihren Wertebereichen

Die erste Phase eines CLP-Programmes ist der Aufbau der Constraintvariablen. Um zu entscheiden wie der Aubau der Constraintvariablen funktionieren soll, ist die Darstellung des Problems als Menge von Constraintvariablen zu wählen.

Ein möglicher Stundenplan besteht aus Blöcken die Teil ihrer Vorlesungen sind, und die zu bestimmten Zeiten an bestimmten Tagen in bestimmten Räumen stattfinden. Ein Block erstreck sich möglicherweise über mehrere dieser Vorlesungseinheiten, und befindet sich über seine gesamte Dauer in einem Raum. Jeder Tag eines Stundenplanes ist N zeitlich gleichlange Intervalle, die Vorlesungseinheiten genannt werden eingeteilt. Weiterhin stehen an jedem Tag und zu jeder Vorlesungseinheit die definierten Räume für Veranstaltungen zur Verfügung. Ein leerer noch zu erstellender Stundenplan kann also auch als eine Liste von "freien Plätzen" im Stundenplan angesehen werden, denen die Vorlesungseinheiten der im Stundenplan zu Verteilenden Blöcke zugewiesen werden. Ein "freier Platz" ist dabei das 3-Tupel T=(Tag,Zeit,Raum).

Das Problem der Erstellung eines Stundenplanes, ist also das Problem der Verteilung aller Vorlesungseinheiten aller Blöcke aller Vorlesungen auf die freien Plätze des Stundenplanes,so daß :

- Die Blockstruktur der Blöcke aller Vorlesungen erhalten bleibt.
- Die allgemeinen Constraints eines Stundenplanes einghalten werden.
- Die spezifischen Constraints des Stundesplanes eingehalten werden.

Die freien Plätze als statische Einheiten anzusehen, denen im Verlauf der Stundenplangenerierung Vorlesungseinheiten zugewiesen werden hat jedoch Nachteile, da die Constraintvariablen dann die Vorlesungseinheiten der Blöcke repräsentieren würden. Die Erstellung der benötigten Constraints über diesen Constraintvariablen wäre unnötig komplex, zumal den die Constraintvariablen für nicht belegte freie Plätze eine "leere Vorlesungseinheit" zugewiesen werden müsste.

Besser ist es Tage,Zeiten und Räume als Constraintvariable anzusehen, und jeder Vorlesungseinheit eines Blockes ein 3-Tupel DTR=(Tag,Zeit,Raum) zuzuweisen. Tag, Zeit und Raum sind dabei die Constraintvariable. Die Liste aller DTR-3-Tupel (also der anfangs noch freien Plätze) ist dann genau so lang wie es zu verteilende Vorlesungseinheiten gibt. Dieser Ansatz hat auch den großen Vorteil, das die Constraintvariable a priori einer Vorlesungseinheit eines bestimmten Blockes einer Vorlesung zugeordnet sind, und sich somit die Constraints des Stundenplanes wesentlich einfacher formulieren lassen.

Um die Blöcke von Vorlesungen eindeutig benennen zu können wird für jeden Block eine eindeutige Block-Identifikation (BID) in Form einer positiven ganzen Zahl vorgesehen. Besteht eine Vorlesung aus mehreren Blöcken, so hat jeder Block eine eigene BID, und auch über die Vorlesungsgrenzen hinweg sind die BID's eindeutig, so daß sich anhand einer BID die Vorlesung bestimmen läßt, zu der ein Block mit einer bestimmten BID gehört.

Um die eindeutige Zuordnung von DTR-3-Tupeln zu ihren Blöcken zu erreichen wird ein DTR-3-Tupel als Prolog-Struktur folgendermaßen definiert dtr/4(Tag,Zeit,Raum,bid) Tag,Zeit und Raum sind dabei Constraintvariable während der 4. Parameter gebunden ist und eine BID darstellt. Eine solche dtr/4 Struktur stellt also einen Platz im Stundenplan dar, der von vornherein einer Vorlesungseinheit eines Blockes mit der BID bid zugewiesen ist. Indem nun während der Labeling-Phase die Constraintvariablen an Werte gebunden werden, wird festgelegt wann und wohin diese Vorlesungseinheit im Stundenplan gelegt wird. Besteht ein Block aus mehreren Vorlesungseinheiten, so werden für ihn so viele dtr/4 Strukturen angelegt wie er Vorlesungseinheiten hat. Da alle dtr/4-Strukturen aller Blöcke zusammen in einer Liste stehen, die während der gesamten Laufzeit nicht umsortiert wird, geschieht die Zuordnung von dtr/4-Strukturen mit einer bid (eines Blockes) zu den einzelnen Vorlesungseinheiten über die Reihenfolge der dtr/4 Strukturen in der Liste. Die erste dtr/4-Struktur eines Blockes in der Liste entspricht der ersten Vorlesungseinheit des Blockes, die zweite Struktur der zweiten Vorlesungseinheit und so fort. Den in der Datenbasis definierten Tagen, Zeiten und Räumen werden in der Reihenfolge ihrer Definition in der Datenbasis beginnend mit 0 Indizes zugeordet, die die Werte der Wertebereiche der Constraintvariablen darstellen.

> **Wahl der Darstellung des Problems :** Der Problemraum wird also als eine Liste von dtr/4(Tag,Zeit,Raum,bid) Strukturen modelliert, wobei die Länge der Liste der Gesamtzahl der im Stundenplan zu verteilenden Vorlesungseinheiten aller Blöcke aller Vorlesungen entspricht. Jede dtr/4-Struktur entspricht dabei einer Vorlesungseinheit eines Blockes (mit der Block-ID BID). Für jeden Block existieren in der Liste so viele dtr/4-Strukturen wie der Block Vorlesungseinheiten hat. Die n-te dtr/4 Struktur eines Blockes in der Liste entspricht dabei der n-ten Vorlesungseinheit dieses Blockes.

Während der Phase der Erstellung der Constraintvariablen wird anhand der Datenbasis die Anzahl N aller zu verplanenden Vorlesungseinheiten berechnet, jedem Block der jeder definierten Vorlesung eine eindeutige Block-ID zugewiesen und schließlich eine Liste von dtr/4(Tag,Zeit,Raum,bid) Constraintvariablenstrukturen der Länge N erstellt.

4.1.2 Erstellung von Constraints über den Constraintvariablen

Die über den Constraintvariablen zu definierenden Constraints lassen sich folgendermaßen einteilen

- Allgemeine Constraints
  - **Constraints bezüglich der Räume.** Da Veranstaltungen nur in Räumen stattfinden dürfen die die von der Veranstaltung benötigten Ressourcen bereitstellen, wird für jede Vorlesung anhand der für diese Vorlesung in der Datenbasis (DB) angegebenen Ressourcen und der Definition der Räume in der DB die Menge der Räume berechnet die für die Vorlesung überhaupt in Frage kommen. Die Wertebereiche aller Raum-Constraintvariable die dieser Vorlesung zugeordnet sind, werden dann auf diese Menge eingeschränkt. Eigene Raum-Constraints, die vom Constraint-Solver der CLP-Laufzeitumgebung überwacht werden müßten, sind so nicht notwendig.
  - **Constraints bezüglich der Zeit.**
    - Es müssen Constraints über den Constraintvariablen aufgebaut werden, die die Anforderung modellieren, daß in einem Raum zu einer Zeit nicht mehr als eine Veranstaltung stattfinden darf. Zu diesem Zweck wird über allen dtr/4(Tag,Zeit,Raum)-Constraintvariablenstrukturen des Problemraumes folgendes lineare Constraint aufgebaut

      $$n \cdot Raum + m \cdot Tag + Zeit = X$$

      m:*Anzahl der Zeiteinheiten pro Tag*
      n:*Anzahl aller Tage pro Woche multipliziert mit m*
      Alle so erhaltenen Constraintvariablen X werden in einer Liste gesammelt. Schließlich werden über den Elementen X dieser Liste Constraints aufgebaut, so daß alle Elemente X paarweise verschieden sind. (Die Paarweise Ungleichheit aller Elemente einer Liste ist unter ECLiPSe mittles des alldifferent/1-Constraints für Listen schon definiert) **Anmerkung:** Dieses Constraint kann wie oben definiert unter Verwendung des ECLiPSe-Constraint-Prädikates #=/2 umgesetzt werden.
    - Weiterhin müssen Constraints über den Constraintvariablen aufgebaut werden, die verhindern, daß sich Vorlesungen mit

nicht disjunkten Hörergruppen bzw. Blöcke ein und derselben Vorlesung zeitlich überlappen. Zu diesem Zweck werden für jede Vorlesung V, alle zu dieser Vorlesung gehörenden dtr/4-Constraintvariablenstrukturen sowie alle dtr/4 - Constraintvariablenstrukturen aller Vorlesungen die nicht zu V parallel stattfinden dürfen in einer Liste gesammelt.Für alle ungeordneten Paare dtr/4-Paare

$$(dtr(Tag1, Zeit1, Raum1), dtr(Tag2, Zeit2, Raum2))$$

dieser Liste wird dann über den Constraintvariablen des Paares folgendes lineare Constraint aufgebaut.

$$m \cdot Tag1 + Zeit1 \neq m \cdot Tag2 + Zeit2$$

m: *Anzahl der Zeiteinheiten pro Tag*
Anmerkung: Dieses Constraint kann wie angegeben unter Verwendug des ECLiPSe-Constraint-Prädikates #\=/2 umgesetzt werden.

- **Constraints bezüglich der Blockstrukturen.** Da während der Stundenplanerstellung ganze Blöcke im Stundenplan verteilt werden müssen, deren Anfangszeiten während des Labeling zu berücksichtigen werden, müssen Constraints über den Constraintvariablen der Blöcke aufgebaut werden, so daß der Zusammenhang der Blöcke im Stundenplan sichergestellt ist. D.h. Zwischen dev Raum-Constraintvariablen eines Blockes ist ein = Constraint aufzubauen, da jeder Block über seine ganze Zeitdauer in einem Raum stattfinden muß. Gleiches gilt für die Tag-Constraintvariablen eines Blockes, da ein Block sich nicht über mehrere Tage erstrecken kann. Hat ein Block n (Indizes $0..n-1$) Vorlesungseinheiten, so sind jeweils zwischen Zeit-Constraintvariablen des Blockes Constraints aufzubauen, so daß folgendes gilt :

$$\forall 0 < i \leq n-1 \bullet Zeit_{i-1} + 1 = Zeit_i$$

mit $i \in \mathcal{N}$, $Zeit_i$ ist dabei die Zeitconstraintvariable der i-1ten Vorlesungseinheit des Blockes. Der Wertebereich der Zeitconstraintvariablen ist so aufsteigend so geordnet, das zwei Werte t und $t+1$ zwei direkt aufeinanderfolgende Zeiteinheiten darstellen.
Die so zwischen den Zeit-Constraintvariablen aufgebauten Constraints sichern den zeitlichen Zusammenhang von Blöcken. Wird

einer der Zeit-Constraintvariablen dann ein Wert aus ihrem Wertebereich zugeordnet, so werden durch Constraint-Propagation auch die anderen Zeiteinheiten des Blockes festgelegt. Treten dabei Inkonsistenzen mit anderen Constraints auf, so schlägt die ursprüngliche Wertzuweisung fehl, so daß Bloecke nur "in einem Stück" im Stundenplan verplant werden können. **Anmerkung :** Die obigen Einschränkungen können mit Ausnahme der Iteration ($\forall$) mittels des ECLiPSe-Constraint-Prädikates $\#=/2$ umgesetzt werden.

- Spezifische Constraints

  - Constraints bezüglich der Vorlesungen.
    Vorlesung_Am Einschränkung aller Tag-Constraintvariable aller dtr/4-Strukturen der Vorlesung auf den definierten Tag
    Vorlesung_Um Einschränkung der Zeit-Constraintvariable der ersten dtr/4 Struktur der Vorlesung in der Problemraumliste auf die definierte Zeit
    Vorlesung_Am_Um Einschränkung der Zeit sowie der Tag Constraintvariablen der ersten dtr/4 Struktur der Vorlesung in der Problemraumliste auf die definierte Zeit, bzw. den definierten Tag.
    Vorlesung_Nicht_Am, Vorlesung_Nicht_Um Ausschluss der definierten Zeiten bzw. Tage aus den entsprechenden Constraintvariablen der Vorlesung.
    Vorlesung_Nicht_Am_Um Über allen dtr/4 Constraintvariablenstrukturen wird folgender Constraint aufgebaut

    $$Tag = AM \Leftrightarrow Zeit \notin UM$$

    Tag,Zeit sind Constraintvariable aus der jeweiligen dtr/4 Struktur
    AM ist der mittels der Einschränkung aus der Datenbasis Vorlesung_Nicht_Am_Um definierte Tag für den die Zeiteinschränkungen gelten
    UM ist die Menge aller Zeiten, zu denen die Vorlesung für die die Einschränkung gilt am Tag AM nicht stattfinden darf.

  - Constraints bezüglich der Dozenten/Assistenten. Für jeden definierten Dozenten werden alle dtr/4-Strukturen (Zeiteinheiten) aller Blöcke aller Vorlesungen an denen der Dozent/Assistent teilnimmt ermittelt. Dann werden die für den Dozenten/Assistenten definierten Zeiteinschränkungen aus der Datenbasis ermittelt, und in interne Constraints auf den für den Dozenten ermittelten dtr/4 - Constraintvariablenstrukturen umgesetzt.

Dozent_Nicht_Am Für diese Einschränkung werden aus den Wertebereichen aller Tag-Constraintvariablen der ermittelten dtr/4 Strukturen die über die Einschränkung definierten Tage entfernt.

Dozent_Nicht_Um Wie Dozent_Nicht_Am nur für Zeiten.

Dozent_Nicht_Am_Um Über allen Zeit - Constraintvariablenstrukturen der ermittelten dtr/4-Strukturen wird folgendes Constraint aufgebaut.

$$Tag = AM \Leftrightarrow Zeit \notin UM$$

Tag,Zeit sind Constraintvariable aus der jeweiligen dtr/4 Struktur

AM ist der mittels der Einschränkung aus der Datenbasis Dozent_Nicht_Am_Um definierte Tag für den die Zeiteinschränkungen gelten

UM ist die Menge aller Zeiten, zu denen der Dozent für den die Einschränkung gilt am Tag AM nicht verfügbar ist.

4.1.3 Labelingphase (Lösungssuche)

Zur Lösungssuche wird eine direkte Heuristik mit einer lokalen Branch-And-Bound - Optimierung kombiniert, wobei partielle Lösungen die zu Inkonsistentenzen führen durch die Constraint-Propagation-Fähigkeiten des CLP-Systems frühzeitig erkannt werden. Die Lösungssuche läuft in zwei Phasen ab

1. Vorsortierung der im Stundenplan zu verteilenden Blöcke. Dies geschieht, indem eine heuristisch geordnete Liste aller Block-ID's erstellt wird. Dieser erste Schritt entspricht der direkten Heuristik. Die Liste ist nach Prioritäten, die für jede Block-ID berechnet werden absteigend sortiert. Die Prioritäten sind folgende

   (a) Zeitlich grössere Blöcke haben eine grössere Priorität als zeitlich kleinere.

(b) Innerhalb der Prioritätsklasse der zeitlichen Größe von Blöcken haben Vorlesungen mit der größeren Kardinalität der Teilnehmergruppe eine höhere Priorität als Blöcke mit einer kleineren Teilnehmergruppe

(c) Innerhalb der Prioritätsklassen aus Punkt (b) haben die Blöcke denen weniger mögliche Räume zur Verfügung stehen eine höhere Priorität als Blöcke deren Anzahl möglicher Räume grösser ist.

2. Verplanung der Blöcke im Stundenplan in der Reihenfolge der direkten Heuristik. In diesem Schritt wird über eine Branch-And-Bound Suche versucht die Kosten für die Verplanung der Blöcke im Stundenplan zu minimieren. Die Branch-And-Bound Suche versucht für jeden Block die optimale Platzierung im Stundenplan zu finden. Die hierfür verwendete Kostenfunktion summiert die Kosten die für den Tag und den Raum definiert sind die einem Block zugewiesen werden sowie die Kosten aller Zeiteinheiten über die sich der Block erstreckt auf. Diese Kosten (Blockfaktoren) werden für jeden Tag, jede Zeiteinheit und jeden Raum in der Datenbasis definiert.

4.1.4 Ausgabe des Ergebnisses

Nach Abschluß der Labeling-Phase sind die Constraintvariable der dtr/4 - Liste an Werte gebunden. Diese Liste stellt also die eigentliche Lösung bzw. den generierten Stundenplan dar. Diese Liste kann nun traversiert und in unterschiedliche Ausgabeformate transformiert werden. Implementiert wurden eine Zeichenorientierte Ausgabe des Stundenplanes als Liste von Terminen für Blöcke, sowie Erstellung einer HTML-Tabelle mit den Stundenplandaten.

4.2 Scheinvorlesungen

Um Einschränkungen wie "Mindestens eine Freistunde zur Mittagszeit für jede Hörergruppe", "Dozent X hat mindestens N freie Tage pro Woche" und "Vorbereitungszeit vor Vorlesung X einplanen" erfüllen zu können, können Scheinvorlesungen (Dummy-Vorlesungen) definiert werden, die in als nicht regulär definierten Räumen (Dummy-Räume) stattfinden. So ist es z.b. möglich eine Dummy-Vorlesung zu definieren, an der nur der Dozent teilnimmt, der Vorbereitngszeit vor einer Vorlesung benötigt, und diese Scheinvorlesung dann so einzuschränken, daß sie direkt vor der Vorlesung stattfindet, vor der der Dozent Vorbereitungszeit benötigt.

Die allgemeine Anforderung an Stundenpläne “Mindestens eine Freistunde zur Mittagszeit für jede Hörergruppe” wird ebenfalls über die Definition von Scheinvorlesungen umgesetzt. Die Definition der entsprechenden “Mittagspausen-Vorlesungen” nimmt das System jedoch automatisch auf Basis der mit group/2 in der Datenbasis definierten Hörergruppen vor. Da es, wie im Fall der Mittagspausen, möglich sein soll mehrere Scheinvorlesungen gleichzeitig in einem Dummy-Raum stattfinden zu lassen, sind für die dtr/4 Strukturen (Zeiteinheiten) von Scheinvorlesungen die Constraints die sicherstellen daß pro Zeit und Raum nicht mehr als eine Vorlesung stattfindet, nicht anzuwenden.

4.3 Ablauf des Stundenplanungssystems

Die erste Phase der Stundenplangenerierung beginnt mit dem Einlesen der in einer eigenen Datei als Prolog-Faktenmenge vorliegenden Datenbasis. (Siehe DEFINITION DER IN DER DATENBASIS VERWENDETEN PRÄDIKATE S.45). Der Name der Datei der Datenbasis ist beim Start der Stundenplangenerierung vorzugeben. Die Datenbasis wird dann in einem ersten Schritt compiliert und anschließend auf Fehler getestet. Sollte kein Fehler vorliegen so ist der nächste Schritt der Aufbau interner Datenstrukturen. Während dieses Schrittes werden auch die Scheinvorlesungen für die Mittagspausen aller Hörergruppen, sowie für die mit vorlesungsfreieTage/2 definierten freien Tage von Dozenten erstellt. Ebenso werden in diesem Schritt die Dummy-Räume *mensa* und *freizeit* dynamisch der Datenbasis hinzugefügt. Dazu werden aus den Informationen aus der Datenbasis für den weiteren Programmablauf benötigte Berechnungen vorgenommen, und die Ergebnisse wiederum in der Datenbasis abgelegt. Nach Abschluß dieses Schrittes folgt der Aufbau der Constraintvariablen. Das Ergebnis ist die oben beschriebene Liste von dtr/4-Strukturen, auf der dann in den nächsten Schritten die Constraints aufgebaut werden (Aufbau von Constraints bezüglich der Räume, Zeiten, Blockstrukturen Ausschluss von mehr als einer Vorlesung pro Raum zur Zeit, Umsetzung aller in der DB definierten spezifischen Constraints in interne Constraints über den jeweils betroffenen Constraintvariablen). An den Aufbau der Constraints schließt sich dann die Labelingphase sowie die Ausgabe des Ergebisses an.

4.4 Definition der in der Datenbasis verwendbaren Prädikate

Mit Liste ist im Folgenden immer eine Liste in Prolog-Syntax gemeint.

tag/2(TagName,BlockFaktor) TagName ist der eindeutige Name des Tages, der auch später während der Ausgabe des Stundenplanes verwendet wird. BlockFaktor wird während der Branch-And-Bound Suche des Suchprozesses von der Kostenfunktion für die Position eines Blockes im Stundenplan verwendet. Die Reihenfolge der Tage ist durch die Reihenfolge der Fakten in der Datenbasis definiert.

tageOhneMittag/1(TageListe) Hiermit werden Tage definiert an denen keine Mittagspausen zu berücksichtigen sind (z.b. Samstags)

zeit/2(ZeitName,BlockFaktor) ZeitName ist der eindeutige Name der Zeiteinheit, der auch später während der Ausgabe des Stundenplanes verwendet wird. BlockFaktor wird während der Branch-And-Bound Suche des Suchprozesses von der Kostenfunktion für die Position eines Blockes im Stundenplan verwendet. Die Reihenfolge der Zeiten am Tag ist durch die Reihenfolge der Fakten in der Datenbasis definiert.

mittagszeit/1(Zeitenliste) Die Zeiten in der Zeitenliste definieren die Zeiteinheiten von denen Hörergruppen mindestens eine Freistunde für das Mittagessen haben müssen.

raum/3(RaumName,Ressourcenliste,BlockFaktor) RaumName ist der eindeutige Names des Raumes, der auch später während der Ausgabe des Stundenplanes verwendet wird. Ressourcenliste ist die Liste der Ressourcen die dieser Raum zur Verfügung stellt. Minimal enthält diese Liste folgende Elemente:

groesse/1 Diese Struktur deren Parameter eine ganze Zahl ist und die Größe des Raumes darstellt muß immer das erste Element der Ressourcenliste sein.

regular Für Räume die nicht mit Hilfe des Prädikates dummyRaum/1 als nicht regulärer Raum definiert werden, ist diese Ressource anzugeben.

Raumkürzel Die Ressourcenliste eines Raumes enthält eine Ressource die den Raum selbst kennzeichnet. So ist es Vorlesungen möglich über die Ressourcenanforderungen die Menge der Räume einzuschränken in denen die Vorlesung stattfinden kann.

dummyRaum/1(RaumName) Räume in denen keine reguläre Vorlesung stattfinden kann, werden benötigt um Scheinvorlesungen (Dummy-

Vvorlesungen) einen Raum zuweisen zu können. Räume die für Scheinvorlesungen verwendet werden, also nicht regulär sind, werden mittels dieses Prädikates als Dummy-Raum gekennzeichnet. RaumName ist dabei der bei der Definition des Raumes angegebene RaumName.

dozent/1(Name) Definiert einen Dozenten/Assistenten mit den Namen Name.

group/2(GruppenName,Groesse) Mittels dieses Prädikates werden Hörergruppen definiert. Als Gruppenname wird kann z.b. die Angabe von Studienrichtung & Fachsemester dienen (z.b. wi7 für Wirtschaftsinformatik 7. Semester). Groesse definiert die Anzahl von Studenten in dieser Gruppe. Es wird davon ausgegangen, daß alle so definierten Gruppen paarweise disjunkt sind.

dummyGroup/1(GruppenName) Es kann sinnvoll sein Gruppen zu definieren die real nicht existieren, deren Groesse bei der Berechnung von Hörerzahlen für Vorlesungen nicht breücksichtigt wird, und die in der Ausgabe von Stundenplänen nicht erscheinen. Mit Hilfe solcher Gruppen läßt sich aber erreichen, daß Vorlesungen mit eigentlich disjunkten Hörergruppen nicht parallel stattfinden können. Wird eine solche Gruppe benötigt, so muß sie mit group/2 definiert und mit dummyGroup/1 unter Angabe des Gruppennamens als Dummy-Gruppe gekennzeichnet werden.

subgroup/4(Obergruppe,Subgruppenname,Kontext,Größe) Mittels dieses Prädikates lassen sich Subgruppen wie in Abschnitt 3.1 Vereinfachtes Subgruppenmodell beschrieben definieren. Obergruppe ist der Name der mit group/2 definierten Obergruppe. Subgruppenname ist der Name der Subgruppe wie er später bei der Definition von Vorlesungen verwendet wird. Kontext ist der Name des Kontext in dem sich diese Subgruppe befindet und Größe ist die Anzahl von Studenten in dieser Subgruppe.

vorlesung/4(Name,Teilnehmerliste,Ressourcenliste,Blockliste) Mit diesem Prädikat werden zu haltende Vorlesungen für den zu generierenden Stundenplan definiert. Nehmen an einer Vorlesung mehrere Dozenten/Assistenten teil, so ist diese Vorlesung nur einmal mit allen entsprechenden Dozenten/Assistenten in der Hörerliste zu definieren.

Name ist der eindeutige Name der Vorlesung wie er auch bei der Ausgabe des Stundenplanes verwendet wird

Teilnehmerlise ist die Liste aller an der Veranstaltung Teilnehmenden Gruppen wie sie mittels group/2 und/oder subgroup/2 und dozent/1 definiert wurden. Auch die die Vorlesung haltenden Dozenten/Assistenten befinden sich in dieser Liste.

Ressourcenliste ist eine Liste von Ressourcen. Die Blöcke der Vorlesung können nur in Räumen stattfinden die mindestens die in dieser Liste angegebenen Ressourcen zur Verfügung stellen. Vorlesungen die nicht mittels dummyVorlesung/1 als Scheinvorlesug definiert wurden müssen mindestens die Ressourcenanforderung *regular* haben, da seine Vorlesung natürlich nur in einem realen Raum stattfinden kann. In der Ressourcenliste ist es möglich Ressourcen anzugeben von denen der Raum in dem die Vorlesung stattfindet nur mindestens eine Ressource zur Verfügung stellen muß. Dies geschieht indem diese Ressourcen als Liste von Ressourcen als Parameter einer oder/1 Struktur in die Ressourcenliste eingetragen werden. Benötigt eine Vorlesung also z.b. die Ressourcen *overhead* oder *videoprojektor* und soll entweder in *Hörsaal hs1* oder *hs2* stattfinden so kann dies durch oder([overhead,videoprojektor]),oder([hs1,hs2]) angegeben werden. Diese Vorlesung findet dann also entweder in hs1 oder 2 statt, die mindestens die Ressourcen overhead oder videoprojektor stellen müssen.

Blockliste Über diese Liste wird die Aufteilung von Vorlesungen in Blöcke vorgenommen. Sie enthält eine Liste von xxx/1 Strukturen wobei der Parameter die Anzahl der durch die Struktur definierten Blöcke angibt und xxx Art und Größe der Blöcke. Dabei kann xxx für die Konkatenation von "single", "block2" oder "block3" (Blöcke mit einem,zwei oder 3 Vorlesungseinheiten) mit dem leeren String "" oder "NoCluster" stehen. NoCluster-Blöcke können nicht gemeinsam mit anderen Blöcken derselben Vorlesung an einem Tag stattfinden. Mögliche Blockstrukturen sind also single/1, singleNoCluster/1, block2/1, block2NoCluster/1, block3/1 und block3NoCluster/1. Beispielsweise definiert die Blockliste [block2NoCluster(2), single(1)] daß die Vorlesung aus insgesamt 5 Vorlesungseinheiten besteht: zwei Blöcke mit jeweils 2 Vorlesungseinheiten und ein Block mit einer Vorlesungseinheit. Die Doppelblöcke dürfen mit keinem anderen

Block derselben Vorlesung an einem Tag stattfinden.

dozentNichtAm/2(Dozent,TageListe) Der angegebene Dozent kann an den in der TageListe angegenenen Tagen keine Vorlesungen halten.

dozentNichtUm/2(Dozent,ZeitenListe) Der angegenene Dozent kann an keinem Tag zu den in der ZeitenListe angegebenen Zeiten Vorlesungen halten

dozentNichtAmUm(Dozent,Tag,ZeitenListe) Der angegebene Dozent kann am angegebenen Tag zu den in der ZeitenListe angegebenen Zeiten keine Vorlesung halten

vorlesungAm/2(Vorlesung,Tag) Alle Blöcke der angegebenen Vorlesung finden am angegebenen Tag statt.

vorlesungUm/2(Vorlesung,Zeit) Nur für Vorlesungen mit nicht mehr als einem Block. Der einzige Block der Vorlesung beginnt mit der angegebenen Zeiteinheit Zeit.

vorlesungAmUm/3(Vorlesung,Tag,Zeit) Nur für Vorlesungen die aus nicht mehr als einem Block bestehen. Hiermit wird die Anfangszeiteinheit Zeit und der Tag des einzigen Blockes der angegebenen Vorlesung festgelegt.

vorlesungNichtAm/2(Vorlesung,TageListe) Schließt die in der TageListe angegebenen Tage für alle Blöcke der angegebenen Vorlesung aus.

vorlesungNichtUm/2(Vorlesung,Zeitenliste) Schließt die in der ZeitenListe angegebenen Zeiteinheiten für alle Blöcke der angegebenen Vorlesung aus.

vorlesungNichtAmUm/3(Vorlesung,Tag,Zeitenliste) Schließt am angegebenen Tag die in der ZeitenListe angegebenen Zeiteinheiten für die angegebene Vorlesung aus.

vorlesungsfreieTage/2(Dozent,N) Führt zur Definition von Scheinvorlesungen, so daß der angegebene Dozent zusätzlich zu den mit dozentNichtAm/2 definierten Tagen mindestens N Vorlesungsfreie Tage pro Woche hat.

vorlesungenNichtZusammenhängend/2(A,B) Legt fest, daß die Vorlesungen A und B im Stundenplan zeitlich nicht direkt zusammenhängen.

vorlesungenHintereinander/2(A,B) Definiert, daß die Vorlesungen A und B in dieser Reihenfolge und zeitlich direkt zusammenhängend im Stundneplan erscheinen.

alleBloeckeAmSelbenTag/1(Vorlesung) Legt fest das alle Blöcke der angegebenen Vorlesung an einem Tag stattfinden.

terminNachAbsprache/1(Vorlesung) Legt fest, daß für die angegebene Vorlesung (Name wie mit vorlesung/4 definiert) nicht im Stundenplan zu berücksichtigen ist. Solche Vorlesungen können der Vollständigkeit halber in die Datenbasis aufgenommen werden, oder weil die verwendete Ausgabeprozedur des Stundenplanes diese Vorlesungen berücksichtigt. (Momentan ist dies nicht der Fall).

5 Implementierung

5.1 Aufteilung des Codes in *.pl Dateien

Das der ECLiPSe-Code des Systems ist auf folgende Dateien verteilt.

Main.pl Enthält die Prädikate die den Stundenplanungsprozess starten und steuern, sowie das Prädikat prepareInternals/0, das den Ablauf des Aufbaus der internen Datenstrukturen steuert.

internalStructures.pl Enthält die Prädikate die zum Aufbau der internen Datenstrukturen sowie der Constraintvariablen benötigt werden, bzw. die die internen Datenstrukturen berechnen, die von prepareInternals/0 in die Datenbasis geschrieben werden.

constraints.pl Enthält die Prädikate die über den Constraintvariablen die verschiedenen allgemeinen Constraints erstellen. (Zeit-,Raum- sowie Blockconstraints).

specificConstraints.pl Enthält die Prädikate die die Datenbasis nach definierten spezifischen Einschränkungen absuchen, und die definierten Einschränkungen in ECLiPSe-Constraints über den Constraintvariablen umsetzen.

search.pl Enthält die Prädikate für beide Phasen der Labeling-Phase. (Direkte Heuristik, BranchAndBound-Suche)

HTMLOutput.pl Enthält Prädikate zur Erstellung und Ausgabe einer Lösung (Stundenplan) als HTML-Tabelle.

DummyLectures.pl Enthält die Prädikate zur automatischen Definition von Dummy-Vorlesungen für die Mittagspausen aller definiertem Hörergruppe sowie Dummy-Vorlesungen für die mit vorlesungsfreieTag/2 in der Datenbasis definierten freien Tage von Dozenten.

checkDB.pl Enthält die Prädikate zur Überprüfung der Datenbasis auf einige Fehler wie z.b. doppelt Vergebene Raumnamen.

5.2 Ablauf des Systems

1. Start des Systems über startGeneration/1(2)

   (a) compilieren aller anderen zum System gehörenden *.pl-Files und der Datenbasis

   (b) Überprüfung der Datenbasis mit Hilfe des Prädikates checkDB/0 aus checkDB.pl
   Tritt hier ein Fehler auf wird der Programmablauf mit einer entsprechenden Fehlermeldung verlassen.

   (c) Aufruf von prepareInternals/0 zum Aufbau interner Datenstrukturen

      i. Siehe Abschnitt 5.3

      ii. prepareDummyLectures/0 Erstellt die Dummy-Vorlesungen und Räume für die Mittagspausenvorlesungen aller Hörergruppen sowie die Dummy-Vorlesungen für die freien Tage von Dozenten.

   (d) Start des eigentlichen CLP-Programmablaufes über das Prädikat startTimetabling/1

      i. Aufbau der Problemvariablen als Liste von dtr/4 Strukturen durch das Prädikat mkVariables/2 auf Basis der internen Datenstruktur blockNumbers/1.

      ii. Erstellung der Constraints die ausschließen daß in einem Raum mehr als eine Vorlesung zur Zeit stattfindet. Dies geschieht durch mkXVariables/2 und dem späteren Aufruf von alldifferent/1 auf der von mkXVariables gelieferten Constraintvariablenliste. Das o.g. Constraint wird nur über dtr/4 Variablenstrukturen von Vorlesungen aufgebaut, die keine Scheinvorlesungen sind. Zu diesem Zweck wird die dtr/4 Constraintvariablenliste vor dem Aufruf von mkXVariables entsprechend durch extractNonDummies/3 aufgeteilt.

iii. Aufruf von roomConstraints/2 zur Einschränkungen der Wertebereiche von Raumvariablen.

iv. Aufruf von timeConstraints/1 zum Aufbau von Constraints die ausschließen, daß sich Vorlesungen mit nicht disjunkten Hörergruppen überschneiden.

v. Aufruf von blockConstraints/1 zum Aufbau von Constraints die den Zusammenhalt von Blöcken als Abfolge von Zeiteinheiten sicherstellen.

vi. Aufruf von applySpecificConstraints/1 zur Auswertung der in der Datenbasis definierten Einschränkungen, und deren Umsetzung in über den Constraintvariablen definierte ECLiPSe-Constraints.

vii. mit dem Aufruf von search/1 mit der dtr/4 - Constraintvariablenliste wird der Labelingprozess gestartet

viii. Ausgabe des Ergenisses.

5.3 Dynamisch erstellte interne Datenstrukturen

Hier werden die von prepareInternals/0 erstellten internen Datenstrukturen beschrieben, die während des Programmlaufs an verschiedenen Stellen des Systems benutzt werden. Das Prädikate prepareInternals/0 führt verschiedene Berechnungen durch deren Ergebnisse während des Programmlaufes öfter benötigt werden, und legt die Ergebnisse als Parameter von dynamischen in die Datenbasis geschriebenen Fakten ab. Folgende Fakten werden von prepareInternals/0 in die Datenbasis geschrieben.

dayNumbers/1,timeNumbers/1,roomNumbers/1 enthalten als Parameter jeweils eine Liste von nn/2(Name,Id) Strukturen, die zur Abbildung von Namen aus der Datenbasis **Name** (wie z.b. Raumnamen) auf ihre Repräsentation als Integer **Id** benutzt werden. Die Prädikate zur Berechnung dieser Listen auf Grund der Datenbasis sind dayNumberList/2,timeNumberList/2,roomNumberList/2.

noOfDays/1,noOFTimes/1,noOfRooms/1 Enthalten als Parameter jeweils eine positive ganze Zahl die die Anzahl der definierten Tage, Zeiten bzw. Räume darstellen.

noOfDaysAndTimes/1,noOfTimesToSchedule/1 Enthalten als Paramerter das Produkt der Anzahl der Tage und der Anzahl der Zeiten, bzw. die Gesamtzahl aller zu verplanenden Zeiteinheiten aller Blöcke.

noOfDummyRooms/1 Ist die Anzahl der definierten Dummy-Räume. Diese Angabe wird von der HTML-Ausgaberoutine benötigt.

noOfCourses/1 Anzahl aller Vorlesungen (incl. Scheinvorlesungen)

preparedCourses/1 Der Parameter ist die Liste aller vorlesung/4 (Vorlesungsname, Gruppen, Ressourcen, Blocks) - Strukturen. Diese Liste entspricht allen nicht als terminNachVereinbarung/1 definierten Vorlesungen (incl. Scheinvorlesungen). Der Ressourcenliste wurde jedoch um die Struktur groesse/1 mit der Gesamthörerzahl der Vorlesung erweitert. Falls aufgrund von Subgruppendefinitionen nicht disjunkte Hörergruppen existieren, die ebenfalls an dieser Vorlsung teilnehmen können müssen, aber in der Gruppenliste nicht erscheinen, so werden diese Gruppen der Gruppenliste hinzugefügt. Die Liste wird vom Prädikat preparedCourseList/1 aufgrund der Datenbasis berechnet.

blockNumbers/1 Der Parameter ist die Liste aller block/3 (BlockID, Vorlesungsname, Blockstruktur) Strukturen, die benutzt wird um Block-ID's auf ihren zugehörigen Vorlesungsnamen abzubilden und umgekehrt. Blockstruktur$\in${single, singleNoCluster, block2, block2NoCluster, block3, block3NoCluster}. Die Liste wird vom Prädikat blockNumberList/2 auf Basis von preparedCourses/1 erstellt.

commonGroups/1 Der Parameter ist eine Liste von nst/2 (VorlesungA, VorlesungB). Diese Liste enthält die Vorlesungspaare die sich zeitlich nicht überlappen dürfen, da die Vorlesungen nicht disjunkte Hörergruppen haben. Diese Liste wird vom Prädikat notTheSameTime/2 auf Basis von preparedCourses/1 berechnet.

blockFactors/1 Der Parameter ist eine Liste von blockFactor/2(Name,Faktor) Paaren. Für alle definierten Räume, Zeiten und Tage sind BlockFaktoren (Kosten des jeweiligen Wertes im Wertebereich einer Constraintvariablen) definiert. Mit Hilfe dieser Liste lassen sich die definierten Blockfaktoren für Tage,Räume und Zeiten anhand des Namens eines

Wertes erhalten. Die Elemente dieser Liste werden jeweils zum Teil von den Prädikaten dayNumberList/2, timeNumberList/2 und roomNumberList/2 berechnet.

6 Bewertung und Ausblick

Auf der Basis der Anforderungen an den Stundenplan des Wintersemesters 1988/99 wurde eine Beispielhafte Datenbasis für das Stundenplansystem erstellt (ws9899.pl).

Der Test des Stundenplanungssystems mit dieser Datenbasis ergab, daß das System in der Lage ist, auf Basis dieser Datenbasis einen durchführbaren Stundenplan zu erstellen. Also einen Stundenplan in dem alle Vorlesungen vollständig verteilt sind, alle Hörergruppen die Möglichkeit haben am Mittagessen teilzunehmen und in dem sich Vorlesungen mit nicht disjunkten Hörergruppen nicht überschneiden. Durch die Möglichkeit der Vergabe von Kosten (Blockfaktoren) für die definierten Tage, Zeiten und Räume ermöglicht es zudem eine Ausbalancierung besonders häufig genutzter und weniger oft benutzter Räume und Zeiten. Jedoch ist wahrscheinlich das Expertenwissen über die Erstellung von Stundenplänen noch nicht vollständig in das erstellte System eingefloßen, da dieses Wissen für den Experten der Stundenplanerstellung nicht leicht formalisierbar ist (Dies ist ein allgemein zu beobachtendes Problem bei der Erfassng von Expertenwissen). Es ist aus diesem Grund davon auszugehen, daß sich das vorliegende System für die Erstellung von Stundenplänen als Basis für manuelle Korrekturen eignet, sich jedoch noch nicht zur vollständig automatisierten Erstellung des Stundenplanes einzusetzen ist. Auch in der vorliegenden Fassung ist das Stundenplanungssystem, speziell für noch nicht mit der Stundenplanung vertraute Benutzer, jedoch sicher eine erhebliche Arbeitserleichterung, auch wenn zur Optimierung mehrere Iterationszyklen der Änderung der Datenbais und Erstellung eines neuen Stundenplanes sowie die eventuelle manuelle Nachbearbeitung notwendig sein solten. Zur weiteren Entwicklung des Systems erscheint es mir notwendig das vorliegende System vorerst parallel zur manuellen Stundenplanerstellung einzusetzen. Dabei könnte ein vom Stundenplanungssystem erstellter Stundenplan als Basis für einen zu erstellenden Stundenplan dienen. Im Verlauf dieser Evaluierungsphase werden sich wahrscheinlich weitere Einschränkungen ergeben die in der Datenbasis definierbar sein müssen, die jedoch aus den Informationen über zu haltende Vorlesungen im WS98/99 nicht extrahierbar waren (Noch nicht erfasstes Expertenwissen). Eine weitere Einschränkung des Stundenplanes mit harten Constraints könnte jedoch dazu führen, daß das System nur noch in der Lage ist eine partielle Lösung zu finden. Es erscheint mir deshalb sinnvoll das System um weiche Constraints die gewichtete Optimierungskriterien zulassen zu erweitern. Ein Ansatz in diese Richtung könnte es sein, die Labelingphase um eine dritte Phase, die Reparaturphase, zu erweitern. D.h. ein erster in Bezug auf die weichen Constraints noch nicht optimierter Stundenplan könnte von den beiden schon existierenden Phasen des Labeling erstellt werden. Dieser Stundenplan könnte dann von einem lokalen Suchalgorithmus wie z.b. dem Simulated Annealing unter Verwendung geeigneter Heuristiken versuchen den Stundenplan so zu reparieren, daß möglichst viele noch nicht erfüllte weiche Constraints (sogenannte Repair-Constraints) erfüllbar werden. Das zur Implementierung verwendete ECLiPSe - System bietet die Definition und Verwendung solcher Repair-Constraints über die *repair-Library* schon an.

Zur besseren Wiederverwendbarkeit von bereits erstellten Datenbasen wäre es wünschenswert die Möglichkeit vorzusehen in einer zweiten Datenbasis Prädikate definierbar zu machen, die schon definierte Fakten der ersten Datenbasis redefinieren.

Das vorliegende System beinhalted noch keine eigenständige Benutzeroberfläche, d.h. die Basis-Daten müssen als Prolog-Fakten in einer Datei vorgegeben werden und zur Laufzeit muß die Oberfläche des ECLiPSe - Systems selbst benutzt werden. Für nicht programmiererfahrene Nutzer des Systems ist dies unter Umständen nicht akzeptabel, so daß die Entwicklung einer eigenständigen Oberfläche wünschenswert wäre.

Literatur

[SAET] Survey on automated Examination Timetabling (ASAP Automated Scheduling Optimisation and Planning Group, University of Nottingham). Ein Überblick über das Thema der automatisierten Erstellung von Stundenplänen. Das vorgestellte System ist ein Planungsystem zur Erstellung von Klausurplänen für Hochschulen, daß als Lösungsansatz Genetische Algorithmen verwendet.

[UTSE] UTSE, Construction of optimum timetbales (University of Athens, Department of Informatics DoI/UoA). Beschreibt das Stundenplanungsproblem des DoI/UoA, und ein auf dem ECLiPSe System aufsetzendes Stundenplanungssystem, daß CLP einsetzt um einen durchführbaren Stundenplan zu definieren, und versucht einen optimalen Stundenplan über eine einer Backtrackingsuche überlagerte BranchAndBound Suche und eine Kostenfunktion für Stundenpläne zu finden. Die Komplexität der beschriebenen Probleminstanz ist jedoch recht gering.

[TCLP] Timetabling in Constraint Logic Programming. (Departamento de Informática, Universidade Nova de Lisboa FCT/UNL). Beschreibt das CLP-basierte Stundenplanungssystem der FCT/UNL.

[CHRT] Constraintbasierte Stundenplanug für Universitäten. (Institut für Informatik, Ludwig-Maximilians-Universität München). Beschreibt das über Constraint-Handling-Rules implementierte Stundenplanungssystem des o.g. Instituts.

[CWIR95] CWI - Report. A survey of automated timetabling (Centrum voor Wiskunde en Informatics, Amsterdam). Ein recht umfangreicher Überblick über verschiedene Probleminstanzen des Stundenplanungsproblems und schon untersuchte Lösngsansätze.

[AIMA95] Artificial Intelligence, a modern approach. Überblick über die KI mit einigen kurzen Zusammenfassungen zum Thema Constraint-Logic-Programming. Aus diesem Buch stammt das 8-Queens Beispiel des Abschnittes über CLP.

[STUT99] Tutorial on Search in ECLiPSe. Tutorial über die Realisierung konstruktiver und lokaler Suchverfahren unter ECLiPSe. (Imperial College London 1999)

[ECRC93] Constraint Logic Programming - An informal introduction (ECRC Gmbh. technical report ECRC-93-5). Gute und nicht allzu theoretische Einführung in das Thema CLP.

[ASAP] Die Automated Scheduling Optimisation and Planning Group an der University of Nottingham beschäftigt sich unter anderem auch mit dem Thema Timetabling (automatische Stundenplanerstellung). Auf den WWW-Seiten der ASAP findet sich u.a. auch eine Liste von Ressourcen und Links zum Thema Timetabling. *http://tawny.cs.nott.ac.uk/ASAP/index.html* (Letzter Zugriff 28.Feb.2000)