# Interaktive, automatische Stundenplanung mittels constraintlogischer Programmierung

Hans-Joachim Goltz
GMD – Forschungszentrum Informationstechnik GmbH
Forschungsinstitut für Rechnerarchitektur und Softwaretechnik
GMD-FIRST, Rudower Chaussee 5, 12489 Berlin
e-mail: goltz@first.gmd.de

## 1 Einleitung

Ein Stundenplan ist eine Zuordnung von Ereignissen zu Zeitpunkten bzw. Zeitintervallen, so daß die geforderten Bedingungen (Constraints) erfüllt sind. Neben den Constraints, die alle erfüllt sein müssen und harte Constraints genannt werden, existieren auch Constraints, die möglichst erfüllt sein sollen und als weiche Constraints bezeichnet werden. Seit langem ist bekannt, daß das Stundenplanungsproblem zu der Klasse der NP-vollständigen Problemen gehört (siehe z.B. [5]). Existierende Softwareprodukte für die Stundenplanung schränken die möglichen erzeugbaren Pläne zu stark ein. Außerdem sind sie nicht flexibel genug, um abweichende Anforderungen integrieren zu können und Besonderheiten zu berücksichtigen, die es bei fast jedem Anwender gibt (siehe z.B. [3]).

Ein Ziel unserer Forschung ist die Entwicklung von Methoden, Verfahren und Konzepten für eine interaktive, automatische Stundenplanung, die in Universitäten und Schulen angewendet werden können. Die automatische Lösungssuche soll dabei so realisiert werden, daß möglichst eine Lösung in relativ kurzer Zeit gefunden wird, falls eine existiert. Für die Realisierung einer effizienten automatischen Lösungssuche ist die Problemmodellierung von entscheidender Bedeutung. Deswegen sind für uns die Entwicklung von Konzepten und Methoden der Modellierung von Problemen der Stundenplanung und Untersuchungen zum Einfluß verschiedener Problemmodellierungen auf die Lösungssuche wichtige Forschungsschwerpunkte.

Die Systeme der Stundenplanung sollen so flexibel sein, daß Besonderheiten der Anwender berücksichtigt und Constraints leicht geändert werden können, falls keine grundsätzliche konzeptionelle Änderung der Stundenplanung erforderlich ist. Wichtiger Bestandteil wird eine automatische heuristische Lösungssuche sein, in der der Anwender interaktiv Einfluß nehmen kann. Der Anwender kann einen Plan bzw. Teilplan aber nur so modifizieren, daß keine harten Constraints verletzt sind.

Die automatische Stundeplanung ist ein aktuelles Forschungsgebiet. Im letzten Jahr fand die 2. Internationale Konferenz zu diesem Thema statt. Für die automatische Stundenplanung existieren verschiedene grundlegende Softwaretechniken und Ansätze (siehe z.B. [10]). Zur Verwirklichung unserer Forschungsziele verwenden wird die constraintlogische Programmierung (CLP). Verschiedene CLP-Ansätze werden beispielsweise in [1, 2, 7, 9] diskutiert. Ein anderer constraintbasierter Ansatz wird in [8] beschrieben.

Im Rahmen unserer Forschungen entwickeln wir ein System für die interaktive, automatische Stundenplanung an der Medizinischen Fakultät Charité der Humboldt Universität zu Berlin. Als Implementationssprache wurde die constraintlogische Programmiersprache CHIP gewählt. Insbesondere haben sich die globalen Constraints, die in CHIP enthalten sind, als sehr vorteilhaft für die Modellierung der Probleme erwiesen. In dieser Arbeit berichten wir über diese Systementwicklung und einigen Forschungsergebnissen, die damit im Zusammenhang stehen.

## 2 Problembeschreibung

Für die Medizinische Fakultät Charité der Humboldt-Universität war bisher die Erzeugung von Stundenplänen für den klinischen Studienabschnitt (sechs Semester) problemreich und mit großem manuellen Aufwand verbunden. In der ersten Projektphase wurde deshalb für diesen Studienabschnitt ein automatisches Planungsssytem entwickelt.

In der medizinischen Ausbildung müssen die Studenten relativ viele Pflichtveranstaltungen absolvieren. Neben den Vorlesungen eines Semesters finden Seminare, Untersuchungskurse und Praktika statt, die in Gruppen bis zu 20 Studenten durchgeführt werden. Dazu werden die Studenten eines Semesters in Seminargruppen unterteilt. Die zwei Standorte der Fakultät befinden sich in verschiedenen Stadtbezirken. Einige Lehrveranstaltungen werden auch in anderen Berliner Kliniken und Krankenhäuser durchgeführt. Weitere wichtige Bedingungen dieses Problems der Stundenplanung sind:

1. Die Lehrveranstaltungen können zu jeder Viertelstunde beginnen und können unterschiedliche, vorgegebene Längen haben.
2. Für jede Lehrveranstaltung existieren zeitliche Einschränkungen bezüglich ihrer Durchführung.
3. Zwei Vorlesungen des gleichen Faches sind an verschiedenen Wochentagen einzuplanen.
4. Einige Lehrveranstaltungen finden nur in bestimmten Wochen statt (u.a. Unterteilung in A- und B-Wochen).
5. Eine Lehrveranstaltung, die eine Seminargruppe zu absolvieren hat, kann zur fast gleichen Zeit auch an verschiedenen Orten angeboten werden. Dies kann auch bei Vorlesungen der Fall sein.
6. Vorlesungen eines Semesters bzw. Lehrveranstaltungen einer Seminargruppe dürfen sich zeitlich nicht überlappen.
7. Für jede Seminargruppe sind die Wege zwischen den möglicherweise verschiedenen Veranstaltungsorten der Lehrveranstaltungen zu berücksichtigen.
8. Den Vorlesungen sind im Stundenplan auch Räume konfliktfrei zuzuordnen. Viele Vorlesungen können dabei nur in bestimmten Räumen stattfinden.
9. Die Anzahl der Seminare, Untersuchungskurse und Praktika, die eine Klinik gleichzeitig durchführen kann, ist beschränkt, wobei die Schranke auch zeitabhängig sein kann.
10. Die Wunschzeiten und Raumwünsche für Vorlesungen sind möglichst zu berücksichtigen.

## 3 Modellierung

Der Constraintlöser über endlichen Domänen von CHIP bildet die Grundlage unserer Problemmodellierung. Die gewählte Modellierung eines Problems der Stundenplanung besitzt einen großen Einfluß auf die Propagationseigenschaften des Constraintlösers und damit auf die Effizienz der Lösungssuche. Als Beispiel betrachten wir die Modellierung der 3. Bedingung *"zwei Vorlesungen eines Faches sind an verschiedenen Tagen einzuplanen"*. Wir setzen dabei voraus, daß ein Plan für 5 Tage zu erstellen ist und daß jeder Tag in 48 Zeiteinheiten ( $4 \times 12$ Viertelstunden ) unterteilt ist.

In [7], in der auch CHIP als Implementationssprache verwendet wurde, wird die Modellierung dieser Bedingung wie folgt vorgeschlagen. Für jede Vorlesung werden die folgenden Domänenvariablen für den Vorlesungsbeginn definiert: `Day in 1..5`, `Hour in 1..48` und `Qh in 1..240` (für jede mögliche Viertelstunde der Woche). Die Beziehungen zwischen diesen Variablen wird durch das Constraint `Qh = 48*(Day-1) + Hour` repräsentiert. Um die genannte Bedingung für zwei Vorlesungen eines Faches zu sichern, wird ein weiteres Constraint verwendet, das ausgedrückt, daß die Werte, die den beiden `Day`-Variablen zugeordnet werden, verschieden sein müssen. Wenn die verschiedenen Variablen nur durch die angegebene Gleichung verbunden sind, ist die Propagation zwischen diesen Variablen einer Vorlesung nicht ausreichend, da in Gleichungen nur die Veränderung von Minimum oder Maximum der möglichen Werte propagiert werden. Falls beispielsweise die eine Vorlesung am Tag 3 eingeplant wird, wird zwar in der `Day`-Variable der anderen Vorlesung dieser Wert gestrichen, aber durch das Gleichungsconstraint wird kein Wert aus der Domäne der entsprechenden `Qh`-Variable entfernt.

In der von uns gewählten Modellierung verwenden wir nur die `Qh`-Variable für den Vorlesungsbeginn. Zwei Vorlesungen eines Faches unterscheiden sich höchstens durch die Dauer. Deswegen legen wir fest, welche der beiden Vorlesungen mindestens einen Tag vor der anderen einzuplanen ist. Falls die Vorlesungsdauer verschieden ist (Werte `d1` und `d2`), wird die Dauer jeweils als Domänenvariable mit den beiden möglichen Werten betrachtet (`D1 in [d1,d2]`, `D2 in [d1,d2]`) und die Gleichung `D1 + D2 = d1 + d2` als Constraint erzeugt. Um auszudrücken, daß die Vorlesung mit der Startzeitvariable `Qh1` mindestens einen Tag vor der Vorlesung mit der Startzeitvariable `Qh2` einzuplanen ist, wird eine Domänenvariable `X in [48,96,144,192]` definiert und die Constraints `Qh1 < X` und `X =< Qh2` erzeugt. Wenn nun die eine Vorlesung am Tag 3 eingeplant wird, erfolgt auch sofort eine Einschränkung der Startzeitvariable für die andere Vorlesung. Durch diese Modellierung der 3. Bedingung konnte eine bessere Propagation gegenüber der in [7] diskutierten Modellierung erzielt werden.

Die Problemmodellierung ist auch abhängig von den Möglichkeiten und Propagationseigenschaften des gewählten Constraintlösers. Die in CHIP enthaltenen globalen Constraints `cumulative` und `diffn` haben sich für unsere Modellierung als sehr wertvoll erwiesen. Für eine Erläuterung des Constraints `cumulative` betrachten wir die folgenden Bedingungen:

> *Die Ereignisse $E_1, E_2, \ldots, E_n$ benötigen gleichartige Ressourcen, wobei zu jedem relevanten Zeitpunkt insgesamt $L$ Ressourcen zur Verfügung stehen; jedes Ereignis $E_i$ benötigt $H_i$ Ressourcen und dauert $D_i$ Zeiteinheiten; gesucht sind die Startzeiten $S_i$ der Ereignisse, so daß zu keinenm Zeitpunkt nicht mehr Ressourcen benötigt werden als zur Verfügung stehen.*

Die mathematische Formulierung dieser Bedingungen ist:

für jedes $k \in [\, min\{S_i\}\,,\ max\{S_i + D_i\} - 1\,]$ gilt:
$\Sigma H_j \ \leq\ L$ für alle $j$ mit $S_j \leq k \leq S_j + D_j - 1$

Diese komplexe Bedingung der Ressourcenbeschränkung kann durch das globale Constraint `cumulative` wie folgt ausgedrückt werden:

$$\texttt{cumulative}([S_1, S_2, \ldots, S_n], [D_1, D_2, \ldots, D_n], [H_1, H_2, \ldots, H_n], L)$$

Die Bedingung 9 der Problembeschreibung kann beispielsweise mit diesem Constraint leicht modelliert werde. Seien `S1,S2,...,Sn` die Domänenvariablen für die Startzeiten der Lehrveranstaltungen, die eine Klinik durchführen muß, `D1,D2,...,Dn` die jeweilige Dauer dieser Lehrveranstaltungen und `Max` die maximale Anzahl von Lehrveranstaltungen, die diese Klinik gleichzeitig durchführen kann. Dann kann die Bedingung 9 durch das folgende Constraint ausgedrückt werden:

```
cumulative([S1,S2,...,Sn],[D1,D2,...,Dn],
           [1,1,...,1],Max)
```

Falls `Max` nicht für alle Zeitpunkte gilt, können für die entsprechenden Zeiten "Dummy"-Einheiten hinzugefügt werden, die gerade die nicht vorhandenen Kapazitäten binden.

Durch das globale Constraint `diffn` können abstrakte Bedingungen der folgenden Art direkt modelliert werden:

> *Eine Liste n-dimensionaler Rechtecke ist überlappungsfrei in einem gegebenen n-dimensionalen Rechteck zu plazieren.*

In einem üblichen Stundenplanungsproblems einer Schule kann beispielsweise durch ein solches Constraint ausgedrückt werden, daß alle Unterrichtsstunden bezüglich der drei "Dimensionen" *Zeit, Raum, Lehrer* überlappungsfrei sein müssen.

Mit Ausnahme der letzten Bedingung der Problembeschreibung können alle Bedingungen direkt als Constraints formuliert werden und vor der Lösungssuche erzeugt werden. Aus der letzten Bedingung ergeben sich weiche Constraints. Innerhalb der Lösungssuche wird versucht, diese weichen Constraints möglichst zu erfüllen.

Der Beginn der Lehrveranstaltungen wird jeweils als Domänenvariable definiert, wobei sich zuerst die Werte der Domänen aus der Transformation der relevanten Zeiten in Einheiten von Viertelstunden ergeben (Bedingung 1). Unter Einbeziehung der Dauer der jeweiligen Lehrveranstaltungen werden diese Domänen entsprechend der 2. Bedingung eingeschränkt. Die Modellierung der Bedingungen 3 und 9 wurde bereits oben diskutiert.

Neben der Zeit werden für jede Lehrveranstaltung auch die Wochennummer und die Parallitätsnummer als Größen betrachtet, die unbekannt sein können und dann als Domänenvariable zu definieren sind (Bedingungen 4 und 5). Für jedes Semester und für jede Gruppe kann dann mittels des globalen Constraints `diffn` die Bedingung 6 formuliert werden, wobei die unbekannten Größen die Dimensionen festlegen. Für das Problem der

Modellierung der Berücksichtigung der verschiedenen Mindestabständen zwischen zwei Lehrveranstaltungen in Abhängigkeit des Veranstaltungsortes (Bedingung 7) wurde zusätzlich das symbolische Constraint `element` verwendet und die verschiedenen Wege als spezielle "Unterrichtseinheiten" modelliert.

Um den Vorlesungen Räume zuzuordnen (Bedingung 8), wird als weitere Unbekannte für jede Vorlesung der Raum als Domänenvariable definiert, wobei eine mögliche Einschränkung der Raumzuordnung bereits in der Definition der Domäne integriert wird. Die konfliktfreie Raumzuordnung wird durch Verwendung des Constraints `diffn` gesichert.

## 4 Lösungssuche

Constraintlöser über endlichen Domänen sind nicht vollständig. Für die Erzeugung einer Lösung ist i. allg. Suche erforderlich, die oft durch *"Labelling"* realisiert wird. Dabei werden schrittweise den Constraintvariablen Werte aus ihren Domänen zugeordnet. Als Modifizierung dieser Suchmethode haben wir in [6] vorgeschlagen, daß die Wertzuweisung durch eine Einschränkung der Domäne der ausgewählten Variable ersetzt wird. Als Grundalgorithmus ergibt sich dann:

```
reducing( [], _ ).
reducing(VarList, InfoList) :-
    select_var(Var,VarList,NewVarList),
    reducing_domain(Var, InfoList),
    reducing(NewVarList, InfoList).
```

Durch `select_var/3` wird die Variable aus der Liste der relevanten Variablen ausgewählt, deren Domäne als nächstes durch `reducing_domain/2` zu reduzieren ist, wobei die Art und Weise der Einschränkung auch von den Domänen der anderen Variablen und weiteren Komponenten (`InfoList`) abhängen kann. Im Falle eines Backtrackings wird die mengentheoretische Differenz zwischen der alten Domäne und der reduzierten Domäne als neue Domäne der entsprechenden Variable betrachtet und wieder die Prozedur `reducing_domain/1` bezüglich dieser Variable angewendet. Nachdem die Domänen aller relevanten Variablen reduziert wurden, muß die Suche noch nicht abgeschlossen sein, obwohl der größte Teil der Suche durch eine solche Folge von Einschränkungen der Domänen erreicht werden kann. Diese Idee der backtrackbaren Domänenreduzierung wurde auch hier verwendet. An vielen Beispielen konnte der signifikante Vorteil dieser Methode nachgewiesen werden.

In der Lösungssuche sind zwei Arten von Nichtdeterminismus enthalten: Auswahl einer Domänenvariable und Auswahl der Domäneneinschränkung für die ausgewählte Variable. Natürlich ist es i. allg. nicht möglich, alle Auswahlmöglichkeiten der Suche zu probieren. Für die Lösungssuche sind folglich Heuristiken erforderlich, die die jeweilige Auswahl unterstützen.

In der von uns gewählten Heuristik für die Variablenauswahl ist die Priorität des zugehörigen Faches entscheidend. Die Ausgangspriorität kann in der Problemdefinition festgelegt werden. Wenn es erforderlich ist, kann diese Priorität während der Lösungssuche auch verändert werden, wobei dabei auch ein Zufallsgenerator eingesetzt wird. In den Heuristiken für die Domänenreduzierung ist die Berücksichtigung der Wünsche integriert. Folglich wird im ersten Schritt versucht, eine Domäne so einzuschränken, daß die Wünsche enthalten sind.

In vielen Fällen unserer Versuche konnte eine Lösung entweder in wenigen Suchschritten gefunden werden oder es wurde eine außerordentlich große Anzahl von Suchschritten benötigt, falls das Problem überhaupt lösbar war. Deswegen wählten wir die folgende grundsätzliche Vorgehensweise bei der Lösungssuche, um trotz einer eventuell ungünstig gewählten Heuristik in akzeptabler Zeit eine Lösung zu finden: die Anzahl der erlaubten Suchschritte wird stark eingeschränkt und es werden verschiedene Heuristiken der Variablenauswahl bzw. der Domänenreduzierung verwendet. Somit wird ein Backtracking über verschiedene Heuristiken durchgeführt.

Neben der vollständigen automatischen Planerzeugung kann die Lösungssuche über eine graphische Repräsentation interaktiv beeinflußt werden. Folgende Möglichkeiten der interaktiven Planerzeugung, die beliebig kombinierbar sind, wurden realisiert:

- Lehrveranstaltungen einzeln einplanen (nur widerspruchsfreie Zeiten können zugeordnet werden),
- markierte Lehrveranstaltungen automatisch einplanen,
- markierte geplante Lehrveranstaltungen ausplanen,

- alle noch nicht geplante Lehrveranstaltungen automatisch einplanen.

Bei der interaktiven Beeinflussung der Lösungssuche erfolgt kein “Backtracking” über Entscheidungen des menschlichen Planers. Dadurch ist natürlich möglich, daß der Anwender bei der interaktiven Planung keinen widerspruchsfreien Plan findet, obwohl die automatische Planung eine Lösung erzeugen kann.

# 5 Implementation

Als Implementationssprache wurde die constraintlogische Programmiersprache CHIP (Version 5.0) der Firma COSYTEC aus Frankreich gewählt. Auch die graphischen Schnittstellen wurden bzw. werden in CHIP implementiert. Neben den bereits erwähnten globalen Constraints erwies sich die objektorientierte Komponente von CHIP besonders vorteilhaft für die Implementation.

Die Problemrepräsentation erfolgt in drei Stufen: Problemdefinition, interne relationale Repräsentation, interne objektorientierte Repräsentation. Für die Problemdefinition wurde eine deklarative Problembeschreibungssprache entwickelt. Alle Komponenten eines Problems der Stundenplanung können dadurch auch ohne graphische Schnittstelle leicht definiert werden. Als Zwischenschritt werden die Problemdefinitionen intern in eine relationale Repräsentation transformiert, wobei auch die Daten aus den Definitionen aufbereitet werden ( z.B. Umwandlung der Zeitangaben ). Im nächsten Transformationsschritt wird dann aus der relationalen Repräsentation die objektorientierte Repräsentation erzeugt. Diese Repräsentation wird für die eigentliche Lösungssuche und die graphischen Darstellungen verwendet. Der Zwischenschritt der Transformation ist wegen der Verarbeitung der möglichen Querbezüge zwischen den Definitionen erforderlich. Die Repräsentation des Zwischenschrittes ist außerdem für die Erzeugung verschiedener Lösungsvarianten vorteilhaft. Wenn verschiedene Varianten betrachtet werden sollen, müssen für jede Variante die entsprechenden Objekte erzeugt werden. Für diese Erzugung kann jeweils das Ergebnis des Zwischenschrittes verwendet werden.

Die Ausgabe der erzeugten Stundenpläne erfolgt als HTML-Dateien. Dadurch sind die Ergebnisse der Planung universell weiter verwendbar.

# 6 Ergebnisse

Die erste Testphase konnte erfolgreich abgeschlossen werden. Die Pläne konnten schnell erzeugt werden, falls nicht Widersprüche (Verletzung harter Constraints) auftraten. Diese Widersprüche konnten aber mit Hilfe der interaktiven Planungsmöglichkeiten relativ schnell lokalisiert werden. Die Konfliktbeseitigung erfolgte dann in enger Zusammenarbeit mit der für den Stundenplan verantwortlichen Mitarbeiterin der Fakultät. Dabei zeigte sich auch, daß Hintergrundwissen sehr wichtig ist. Die erzeugten Pläne wurden als HTML-Datei ausgegeben und sind im Internet unter `http://www.first.gmd.de/plan/charite` zu finden.

Seitens der Fakultät wurde die automatisierte Stundenplanung sehr positiv beurteilt. Die Vorteile einer interaktiven, automatischen Stundenplanerzeugung konnten eindeutig nachgewiesen werden. Gegenüber der Situation in den Vorjahren sind die Pläne wesentlich eher verfügbar. Die Zuordnung der Studenten zu Seminargruppen, bei der die Wünsche der Studenten möglichst berücksichtigt werden, kann nun noch zum Ende des vorhergehenden Semesters erfolgen.

# 7 Ausblick

Der ersten Testeinsatz unseres Systems hat bewiesen, daß wir die richtigen Methoden verwenden. Durch diesen Testeinsatz gewannen wir auch wertvolle Informationen für unsere weitere Arbeit. So zeigte sich beispielsweise, welche Bedeutung die interaktive Einflußnahme auf die Suche hat und welche Anforderungen an sie gestellt werden. Die weiteren Schritte der Systementwicklung beinhalten u.a.: Stundenplanung für alle Semester, Raumzuordnung für alle Lehrveranstaltungen, Erweiterung und Modifizierung der Problembeschreibungsbesprache, Modifikationen der erforderlichen Transformationen, Verbesserungen der interaktiven Suche und Vervollständigung der graphischen Schnittstellen. Um die vorgesehene Systementwicklung erfolgreich abzuschließen sind weitere Forschungsarbeiten erforderlich. Zu den Schwerpunkten dieser Forschung gehören Untersuchungen zur Verbesserung der Problemmodellierung und Constraintpropation, und die Entwicklung und der Vergleich verschiedener Methoden und Verfahren der heuristischen Lösungssuche.

# Literatur


[1] F. Azevedo and P. Barahona. Timetabling in constraint logic programming. In *Proc. World Congress on Expert Systems*, 1994.

[2] P. Boizumault, Y. Delon, and L. Peridy. Planning exams using constraint logic programming. In *[11]*, pages 79–93, 1994.

[3] E. Burke, D. Eiiliman, P. Ford, and R. Weare. Examination timetabling in British universities: A survey. In *[4]*, pages 76–90, 1996.

[4] E. Burke and P. Ross, editors. *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York, 1996.

[5] T. B. Cooper and J. H. Kingston. The complexity of timetable construction problems. In *[4]*, pages 183–295, 1996.

[6] H.-J. Goltz. Reducing domains for search in CLP(FD) and its application to job-shop scheduling. In U. Montanari and F. Rossi, editors, *Principles and Practice of Constraint Programming – CP'95*, volume 976 of *Lecture Notes in Computer Science*, pages 549–562, Berlin, Heidelberg, 1995. Springer-Verlag.

[7] C. Guéret, N. Jussien, P. Boizumault, and C. Prins. Building university timetables using constraint logic programming. In *[4]*, pages 130–145, 1996.

[8] M. Henz and J. Würtz. Using Oz for college timetabling. In *[4]*, pages 162–177, 1996.

[9] G. Lajos. Complete university modular timetabling using constraint logic programming. In *[4]*, pages 146–161, 1996.

[10] A. Schaerf. A survey of automated timetabling. Technical Report CS-R9567, Centrum voor Wiskunde en Informatica, 1995.

[11] L. Sterling, editor. *Proc. Internat. Conf. on the Practical Application of Prolog*. London, 1994.