Anwendungen Constraintbasierter Programmierung

Thom Frühwirth und Slim Abdennadher

Institut für Informatik, Ludwig-Maximilians-Universität (LMU)
Oettingenstraße 67, 80538 München
{Thom.Fruehwirth,Slim.Abdennadher}@informatik.uni-muenchen.de

Zusammenfassung Die constraintbasierte Programmierung ist für viele eine der spannendsten Entwicklungen in der Computeranwendung in den letzten zehn Jahren. Dieses junge Gebiet hat von Anfang an in Forschung und Praxis gleichermaßen für Aktivität gesorgt. Kein Wunder, daß sich damit Forscher begeistern und ebenso Millionen verdienen lassen - handelt es sich bei der Constrainttechnologie doch um eine allgemeine Methode für elegantes, effektives, deklaratives Problemlösen mit höheren Programmiersprachen.
In diesem Überblicksartikel stellen wir die contraintbasierte Programmierung anhand der Constraintlogikprogrammierung vor. Wir gehen dabei auf kommerzielle Anwendungen und eigene Arbeiten ein.

1 Einleitung

Constraints[1] eignen sich zur Darstellung von unvollständiger Information, zur Beschreibung der Eigenschaften und Beziehungen von teilweise unbekannten Objekten. Als sehr allgemeiner und abstrakter Begriff haben Constraints die verschiedensten Ausprägungen und Arten, doch sie alle haben wichtige Gemeinsamkeiten.

Ein Beispiel aus dem Alltagsleben: Unser Fahrrad hat ein Zahlenschloß. Wir können uns nicht mehr an die erste Zahl erinnern. Wir wissen nur mehr: Sie ist ungerade, natürlich einstellig und außerdem keine Primzahl. Indem wir die unvollständigen Informationen über die Zahl kombinieren, können wir die gesuchte Zahl, nämlich 9, ermitteln. Dabei sind *ungerade, einstellig* und *keine Primzahl* die Constraints, die die Zahl beschreiben. Man beachte, daß das Constraint *ungerade* für sich allein eine unendliche Menge von Lösungen besitzt. Im allgemeinen reichen Constraints allein nicht aus, um ein Problem vollständig zu lösen. Man muß zwischendurch immer wieder suchen. Hätte uns bei diesem Beispiel die letzte Information gefehlt, so wären wir darauf angewiesen gewesen, die Zahlen 1, 3, 5, 7 und 9 auszuprobieren.

Unter dem constraintbasierten Ansatz versteht man das Lösen von Problemen, indem man die Constraints angibt und löst, die von einer Lösung erfüllt werden müssen. Dabei können zwar spezielle Verfahren zum Einsatz kommen, sie müssen aber gewissen allgemeinen Prinzipien gehorchen. Damit Constraintlösen

[1] Das englische Wort bedeutet (Rand-, Neben-, Wert-)Bedingung, Einschränkung.

in Computerprogramme integriert werden kann und damit Constraints und ihre Lösungen einen Einfluß auf den Ablauf von Programmen haben können, muß es einen Programmteil geben, das die Constraints verwaltet und löst. Dies ist der Constraintlöser. Bei arithmetischen linearen Constraints könnte der Constraintlöser das Gaußsche Eliminationsverfahren anwenden, um z. B. folgende Constraints zu lösen: `X-Y=3, X+Y=7`. Man möchte, daß ein Constraintlöser die Gleichungen möglichst so weit vereinfacht, daß die Wertebelegungen der Variablen explizit werden: `X=5, Y=2`.

Mit der Einbettung von Constraintlösern in Logikprogrammiersprachen (z. B. Prolog) wurde es möglich, schnell und elegant komplexe Probleme durch eine Verbindung aus Constraintbehandlung und Suche zu lösen. Constraintbasierte Programmierung kann vorteilhaft eingesetzt werden zum Schließen mit unvollständiger, ungenauer bzw. unsicherer als auch vollständiger Information (z. B. Finanzanalyse) und zum Lösen kombinatorischer Suchprobleme (z. B. Zeitplanung) in Entscheidungsunterstützungssystemen (auch: Expertensysteme, intelligente Agenten). Seit Anfang der neunziger Jahre wird constraintbasierte Programmierung mit großem Erfolg von mehreren Firmen weltweit kommerziell verwertet, ihr gemeinsamer Umsatz wurde 1996 auf 100 Millionen Dollar geschätzt.

Inhaltsübersicht. Wir geben zuerst einen kurzen Abriß über die Entwicklung der constraintbasierten Programmierung, bevor wir das kommerzielle Potential der Constrainttechnologie darstellen. Schließlich stellen wir zwei konkrete innovative Anwendungen vor, die mit unserer Spracherweiterung Constraint Handling Rules leicht verwirklichbar waren.

2 Constraintlogikprogrammierung

Der Begriff „Constraintlogikprogrammierung“ (CLP) bezeichnet eine Familie von Programmiersprachen, die Ende der achtziger Jahre als eine natürliche Fusion zweier deklarativen Paradigmen entstand, von Constraintlösen und Logikprogrammierung (Abb. 1).

Die Idee der Logikprogrammierung (LP) [14] ist, Probleme logisch zu beschreiben. In diesen Programmiersprachen (z. B. Prolog) werden das zum Problem gehörige allgemeine Wissen und die konkreten Annahmen durch Regeln und Fakten, d. h. eine eingeschränkte Klasse von logischen Formeln ausgedrückt. Durch ihre abstrakte deklarative Natur eignen sich LP-Sprachen gut für die schnelle Erst- und Weiterentwicklung von Prototypen auf der Basis unvollständiger Spezifikationen (Rapid Prototyping).

Die Idee der Constraintlogikprogrammierung (CLP) [15, 11, 4] ist, daß gewisse logische Prädikate als Constraints deklarativ und effizient durch spezielle Algorithmen behandelt werden können. Das heißt, daß die allgemeine Methode der LP-Sprachen, die die Tiefensuche mit chronologischem Rücksetzen (engl. backtracking) verwendet, um spezielle, deterministische Methoden erweitert wird. Dabei erlauben es die Constraints bei der Lösung kombinatorischer Suchprobleme, von vorneherein inkompatible Kombination von der Suche auszuschließen

1970	U. Montanari, Constraintnetzwerke
1970	R.E. Fikes, REF-ARF, Sprache für ganzzahlige, lineare Gleichungen
1972	A. Colmerauer, Marseille, sowie R.A. Kowalski, London, Prolog
1977	A.K. Mackworth, Constraints Netzwerk Algorithmen
1978	J.-L. Lauriere, Alice, Sprache für kombinatorische Suchprobleme
1979	A. Borning, Thinglab, interaktives Graphiksystem
1980	G. L. Steele, Constraints, erste constraintbasierte Sprache, in LISP
1982	A. Colmerauer, Prolog II, Prolog mit Gleichheitsconstraints
1987	H. Ait-Kaci, Austin, Life, Prologerweiterung mit Gleichheitsconstraints
1987	J. Jaffar und J.L. Lassez, CLP(X) - Schema
1987	J. Jaffar, CLP(R), Monash Univ. Melbourne, arithmetische Constraints
1988	P. v. Hentenryck, CHIP, ECRC München, endliche Wertebereiche
1988	P. Voda, Vancouver, Trilogy, CLP-ähnlich mit Ganzzahlarithmetik
1988	W. Older, Ottawa, Bell-Northern Research, Intervallarithmetik
1988	A. Aiba, Tokyo, ICOT, nichtlineare Gleichungssysteme
1988	W. Leler, Termersetzungs-Sprache zum Schreiben von Constraints
1988	A. Colmerauer, Prolog III, Univ. Marseille, Constraints über Listen

Abbildung1. Anfänge der Constraintbasierten Programmierung

und so die Effizienz zu steigern. CLP-Sprachen können als Verallgemeinerung von LP-Sprachen aufgefaßt werden.

Bereits Colmerauers LP-Sprache Prolog II von 1982 erweiterte die Unifikation um die Behandlung von unendlichen, zyklischen Termen (engl. rational trees) im Sinne von Gleichheitsconstraints. Aus diesen Entwicklungen heraus entstanden dann in der zweiten Hälfte der achtziger Jahre die ersten CLP-Sprachen, nämlich CLP(R), CHIP und Prolog III.

CLP(R) bot erstmals eine saubere, deklarative Lösung für die Behandlung von arithmetischen Ausdrücken in LP-Sprachen durch die Einführung von Gleichungen zwischen linearen arithmetischen Ausdrücken über Fließkommazahlen. In Prolog III gab es unter anderem auch lineare Gleichungen - im Gegensatz zu CLP(R) aber erstmals über rationalen Zahlen (Brüchen). Die Constraintlöser dieser CLP-Sprachen basieren dabei auf einem adaptierten Simplexverfahren.

In CHIP wurden erstmals Constraints über endlichen Wertebereichen (Aufzählungen), wie sie aus der Künstlichen Intelligenz bekannt waren, und auch Constraints über Boolescher Algebra, in eine LP-Sprache integriert, um den Suchaufwand für kombinatorische Probleme zu verkleinern.

Wir können in diesen CLP-Sprachen nun etwa `X-Y=3, X+Y=7` schreiben und erhalten die Lösung `X=5, Y=2`. Analog behandelt man nicht nur die Gleichheit, sondern auch Konjunktionen von bestimmten anderen Relationen speziell als Constraints. Zum Beispiel die Ordnungsrelation `<`: Ein Constraint `X<Y, Y<X` hat keine Lösung (und dazu braucht man nicht zu wissen, welche Werte die beiden Variablen annehmen).

Ein vereinheitlichendes Modell für eine formal-logische Beschreibung von CLP, das CLP-Schema, stellten Jaffar und Lassez in [10] vor. Das CLP-Schema ist eine Erweiterung von LP um Constraints, wobei man die positiven theoretischen Eigenschaften von LP möglichst beibehalten hat. Constraints werden als spezielle Prädikate aufgefaßt. Ein allgemeineres Schema wird durch Höhefeld und Smolka in [8] angegeben. Der Hauptunterschied zwischen diesen beiden Schemata liegt in der Vielfältigkeit der Constraints.

In einer deklarativen Programmiersprache soll formal betrachtet einerseits alles, was wir berechnen können, auch logisch aus dem Programm folgen (Korrektheit), andererseits soll alles, was folgt, auch berechenbar sein (Vollständigkeit). Diese Übereinstimmung ist für CLP-Sprachen gegeben [10, 12] und ist einer der Gründe für die Attraktivität von CLP.

Beispiel. Dieses klassische CLP(R)-Beispiel ist aus dem Gebiet des Finanzwesens, genauer gesagt geht es um Zinseszinsrechnung. Es demonstriert eindrucksvoll die Mächtigkeit von CLP-Sprachen. Das Prädikat `mortgage` beschreibt die Beziehungen zwischen Darlehenshöhe, Rückzahlungsrate, Zins und Restschuld bei einer Kreditaufnahme mittels zweier Regeln („`:-`" wird als „wenn" gelesen und „`,`" als „und"). Dabei steht die Variable `D` für die Darlehenshöhe, `T` für die Dauer der Rückzahlung in Monaten, `Z` für den monatlichen Zinssatz, `R` für die monatliche Rückzahlungsrate, und `S` für die Restschuld nach `T` Monaten:

```
mortgage(D, T, Z, R, S) :- T = 0, D = S.
mortgage(D, T, Z, R, S) :- T > 0, T1 = T - 1, D1 = D + D*Z - R,
                           mortgage(D1, T1, Z, R, S).
```

Die erste Regel besagt, daß `mortgage(D, T, Z, R, S)` gilt, wenn `T=0` und `D=S` ist. Die zweite Regel trifft zu, wenn `T>0` ist und verwendet eine Rekursion, um das Problem auf die Zeitdauer `T-1` zurückzuführen.

Die Anfrage `mortgage(100000,360,0.01,1025,S)` liefert die Antwort S=`12625.90`. In Worten: Wenn man 30 Jahre lang ein Darlehen von `100000` mit monatlich `1025` zu einem Monatszins von `1%` zurückzahlt, dann bleibt noch eine Restschuld von `12625.90`. Man kann sich nun fragen, welchen Betrag man in diesem Zeitraum zu diesen Konditionen vollständig zurückzahlen kann. In CLP ist das kein Problem, weil man im Gegensatz zu herkömmlichen Programmiersprachen mit Constraints auch „rückwärts" rechnen kann: Dann liefert die Anfrage `mortgage(D,360,0.01,1025,0)` die Antwort `D=99648.79`, einen nur geringfügig niedrigeren Betrag. Das Berechnungsbeispiel demonstriert eindrucksvoll den Zinseszinseffekt.

Umgekehrt können wir uns fragen, wieviel Monate lang wir die `100000` zurückzahlen müssen. Die Restschuld `S` muß dann gleich oder kleiner Null sein. Die Anfrage `S=<0, mortgage(100000,T,0.01,1025,S)` liefert T=`374` (etwa mehr als 31 Jahre) und S=`-807.96` (d. h. man müßte im letzten Monat nicht mehr die volle Rate zahlen). Wie verhalten sich allgemein Darlehenshöhe und Rückzahlungsbetrag zueinander, wenn man zu obigen Konditionen nach 30 Jahren schuldenfrei sein will? Die Anfrage `mortgage(D,360,0.01,R,0)` liefert die intensionale Antwort `R = 0.0102861198*D`, d. h. das Darlehen beträgt knapp weniger als das Hundertfache der Monatsrate.

3 Kommerzielle Anwendungen

Seit Anfang der neunziger Jahre wird constraintbasierte Programmierung mit Erfolg von mehreren Firmen (Ilog mit IlogSolver und IlogSchedule, Cosytec mit CHIP 4, Siemens Nixdorf mit IFProlog, Prologia mit Prolog IV) weltweit kommerziell eingesetzt. Die Zahl der kommerziellen Anwendungen wurde 1996 auf 300 geschätzt, der Umsatz mit Constrainttechnologie auf etwa 100 Millionen Dollar, mit steigender Tendenz[2] [16]. Die erwähnten Firmen haben jeweils mehrere hundert Kunden für ihre constraintbasierten Produkte gefunden. Während Frankreich, England, USA und Asien stark wachsende Märkte für constraintbasierte Entscheidungshilfesysteme sind, ist der Markt in Deutschland erst im Entstehen.

Der Vorteil des Einsatzes von constraintbasierter Programmierung sind

- die deklarative Modellierung von Problemen mithilfe passender Constraintsysteme, die schneller zu robuster, flexibler und wartbarer Software führt,
- die Möglichkeit zur Darstellung unvollständiger, spärlicher als auch vollständiger Information durch Constraints, die es ermöglicht auch mit ungenauen, unsicheren und unscharfen Daten korrekt zu arbeiten,
- die automatische Propagierung der Effekte, wenn neue Information (in Form von Constraints) bekannt wird, z. B. die Berechnung der Konsequenzen einer Entscheidung und
- die gute Kombinierbarkeit von Constraintlösen mit Such- und Optimierungsverfahren zur Lösung kombinatorischer Probleme, vor allem in Constraintlogikprogrammiersprachen.

Diese vielfältige Flexibilität des constraintbasierten Ansatzes macht den Hauptvorteil gegenüber hochspezialisierten Werkzeugen aus, die u. U. nicht an veränderte Problemstellungen angepaßt werden können. Dafür muß man bei Constraints unter Umständen leichte Abstriche in der Effizienz in Kauf nehmen.

Die Constrainttechnologie ist soweit gereift, daß Constraintlöser und Suchverfahren nicht nur in Logikprogrammiersprachen sondern zunehmend auch als Softwarekomponenten (auch: Bibliotheken) für Standardprogrammiersprachen wie C++ angeboten werden (z. B. von Ilog). Constraintbasierte Software kann vorteilhaft eingesetzt werden zum Schließen mit unvollständiger als auch vollständiger Information (z. B. Finanzanalyse) und zum Lösen kombinatorischer Probleme (z. B. Zeitplanung) in Modellierungs-, Simulations- und Entscheidungsunterstützungssystemen (Expertensysteme, engl. decision support systems).

Constraints werden in solchen Systemen dazu verwendet, um ungenaue und unvollständige Informationen bzw. Daten zu repräsentieren und mit ihnen zu rechnen. Der zugehörige Constraintlöser propagiert Wertebereichsänderungen von einer Variablen zur nächsten. Damit lassen sich die Effekte der Änderung eines Parameters auf alle anderen Parameter in Verallgemeinerung einer Tabellenkalkulation (engl. Spreadsheet) unmittelbar berechnen, sichtbar machen und

[2] Zum Vergleich: Der Umsatz von „Data Mining“ betrug 1996 120 Mill. Dollar.

zur Entscheidungsfindung studieren. Im Unterschied zu einer Tabellenkalkulation kann man mit Constraints aber in beliebige Richtungen rechnen und dies mit ungenauen und unvollständigen Angaben bzw. Daten.

Wissenschaftlich betrachtet sind es vor allem Anwendungen und Fragestellungen aus der Künstlichen Intelligenz, die mit Constraints gut bearbeitet werden können: Computerunterstütztes Sehen (engl. machine vision), Linguistik, Sprachverarbeitung (engl. natural language understanding), zeitliches und räumliches Schließen (engl. temporal and spatial reasoning) und Theorembeweisen.

Hauptsächliche Anwendungsgebiete der Constrainttechnologie sind branchenunabhängig: Zum einen die Modellierung, (ausführbare) Spezifikation, Design, Synthese, Simulation, Verifikation, Fehlerdiagnose von elektronischen, elektrischen und mechanischen Komponenten und ganzen industriellen Abläufen, von Computer-Hardware und Softwarekomponenten und zum anderen Produktions-, Personal-, Finanz-, Verkehrs-, Netzwerk- und Ressourcenplanung, -logistik und -management (insbesondere Zeit- und Kapazitätswirtschaft), sowie Transport- und Plazierungsoptimierung, Design, Konfiguration und Layoutgenerierung.

Aus den oben genannten Gebieten seien hier einige konkrete kommerzielle Anwendungen kurz beschrieben:

- Das System DAYSY von Cosytec adaptiert für die Lufthansa den Einsatz von Personal nach Störungen im Flugbetrieb (Verspätungen, Erkrankung,...), sodaß die Änderungen im Personalplan und die Kosten minimiert werden.
- Nokia Mobile Phones, der zweitgrößte Mobiltelefonhersteller der Welt, verwendet IFProlog zur automatischen Konfiguration von Software für Mobiltelefone.
- Siemens verwendet betriebsintern das in IFProlog mit Booleschen Constraints entwickelte „Circuit Verification Environment“ (CVE2) zum Design und zur Verifizierung von Hardware (VLSI Chips).
- ICL hat mit DecisionPower (ein CHIP-Derivat) bereits 1991 eine Plazierungsanwendung für Hongkong International Terminals, einem der größten Containerhafen der Welt, zur optimalen Plazierung von Containern in Lagerhallen zwischen Ankunft und Abfertigung entwickelt.
- Renault setzt ein CHIP-Derivat seit 1995 im „Short Term Production Planning“ ein, zur optimalen Planung von Zulieferung und Fertigung von Varianten eines Autotyps innerhalb eines Fertigungsabschnittes (engl. workshop).
- Daussault's Anwendung „Made“ in CHIP entscheidet, wo und wie komplexe Flugzeugteile aus einem Blech herausgeschnitten werden sollen, sodaß möglichst wenig Abfall und Zeitverlust entsteht.
- Ilog hat für die französischen Eisenbahnen (SNCF) das Werkzeug „Sagitaire“ entwickelt, das im Bereich des Bahnhofes Paris Nord für über 1700 Züge täglich plant, auf welchen Teilstrecken und Gleisen sie fahren sollen, ohne sich gegenseitig zu behindern.

4 Constraint Handling Rules

Die Erfahrungen mit praktischen Anwendungen zeigen, daß oftmals kein homogenes Constraintproblem vorliegt, sondern eine subtile Kombination verschiedenster Constraintsysteme. Oft treten auch neuartige Constraints auf, die nur mit viel Aufwand in existierende Constraints übersetzt werden können. Häufig ist nach der Übersetzung das Vereinfachungsverhalten der entstehenden Constraints schwächer, als es bei direkter Verwendung eines Constraintlösers für die ursprünglichen Constraints möglich wäre.

Um Constraints so verwenden zu können, wie sie in einer Anwendung auftreten, wurde eine spezielle Sprache, Constraint Handling Rules (CHR), zum Implementieren von Constraintlösern in den 90er Jahren entwickelt [3]. Die Sprache ermöglicht die schnelle Erstellung von Prototypen, von Erweiterungen, von Spezialisierungen und von Kombinationen von Constraintlösern.

CHR sind eine Spracherweiterung, die in eine Programmiersprache als Bibliothek eingebettet werden können. In der Basissprache werden Programme geschrieben, die dann die in CHR implementierten Constraints benutzen können. Andrerseits können CHR-Programme in der Basissprache implementierte Prozeduren als Hilfsprozeduren verwenden. Wir erwähnen nun zwei innovative Anwendungsstudien, die wir mit CHR durchgeführt haben. Beide Studien wurden vollständig in der Constraintlogikprogrammiersprache ECL^{i}PS^e mithilfe ihrer CHR-Bibliothek geschrieben, am European Computer-Industry Research Center (ECRC) in München zusammen mit Projektpartnern.

Der Münchener Mietspiegel Online Der Mietspiegelberechnungsdienst im Internet (IMS)[3] [5] ist ein Beispiel für ein constraintbasiertes Entscheidungsunterstützungssystem. Er erlaubt es, durch eine Formularseite im World Wide Web in wenigen Minuten die ortsübliche Vergleichsmiete einer Wohnung zu berechnen. Vergleichsmieten sind in Rechtsstreitigkeiten als Beweismittel zugelassen.

Die Berechnung basiert auf Größe, Alter, Lage der Wohnung und einer Reihe von detaillierten Fragen über die Wohnung und das Haus, die aus einer statistischen Erhebung als relevant hervorgingen. Man benötigt Angaben, über die man in der Regel nur ungefähr Bescheid weiß, z. B. das Jahr der letzten Renovierung des Hauses oder die exakte Höhe der Kachelung im Bad einer Wohnung. Wegen dieses statistischen Ansatzes tritt eine inhärente Unschärfe auf, die in der Papierversion eines Mietspiegels meist nicht ausreichend berücksichtigt werden kann.

Die elektronische Version des Mietspiegels im Internet kann den Zeitaufwand für den Benutzer von Stunden auf wenige Minuten reduzieren. Mittels Constraints ist der IMS sogar in der Lage, mit ungenauen und unvollständigen Angaben und der statistischen Unschärfe korrekt umzugehen. Damit läßt sich erstmals ein Mietspiegel auch dazu verwenden, bei der Wohnungsuche das Mietpreisniveau zu bestimmen, ohne sich auf eine bestimmte Wohnung festlegen zu müssen.

---

[3] Siehe `http://www.pst.informatik.uni-muenchen.de/personen/fruehwir/`

Der IMS wurde seit 1996 von über zehntausend Benutzern frequentiert. Der IMS gewann im gleichen Jahr den Preis für die beste Anwendung auf der JFPLC Konferenz in Clermont Ferrand, Frankreich und wurde im gleichen Jahr auf der Systems 96 Computermesse in München mit großem Medienecho präsentiert. Er ist das erste System weltweit, das Constrainttechnologie im Internet einsetzt. Der spezialisierte Webserver, der diesen Dienst ins Internet bringt, wurde vollständig in der CLP-Sprache geschrieben: Er nimmt die Benutzerangaben aus dem Formular entgegen und leitet das Berechnungsergebnis an den Benutzer zurück.

Unser Ansatz war, zuerst die Tabellen, Regeln und Formeln der Papierversion des Mietspiegels unter der Annahme zu programmieren, daß alle notwendigen Angaben zu Verfügung stehen. Wegen der Deklarativität von CLP war die Implementierung[4] in wenigen Wochen möglich: Tabellen lassen sich durch Fakten, Regeln durch CLP-Regeln darstellen. Dann haben wir Constraints eingeführt, um die Unschärfe (wegen des statistischen Modells) und Unvollständigkeit (wegen ungenauer oder unvollständiger Benutzerangaben) zu berücksichtigen. Dabei werden die Formeln zur Vergleichsmietenberechnung nun als Constraints betrachtet, der Rest des Programms blieb praktisch unverändert. Das Verhalten dieser Constraints wurde in einem Constraintlöser spezifiert. Es genügte, einen existierenden Constraintlöser aus der CHR-Bibliothek zu erweitern. Die Verwendung einer Constraintlogikprogrammiersprache ermöglicht auch eine einfache Wartung und Anpassung des Programms an neue Mietspiegel.

Optimale Planung von drahtlosen Kommunikationssystemen Mit der Einführung eines Europäischen Standards für digitale kabellose Telekommunikation, DECT, sind kleinräumige Kommunikationsnetze hoher Qualität möglich geworden. Mit kabellosen Sytemen kann das interne Telefonnetz einer Firma jederzeit um Teilnehmer erweitert werden, und dies ohne aufwendige Montagearbeiten. Zudem bringt die digitale Datenübertragung eine verbesserte Übertragungsqualität bei mehr Abhörschutz. Allein in Westeuropa sollen 1999 14 Millionen schnurlose Telefone nach dem DECT Standard verkauft werden, etwa ein Drittel davon in Deutschland.

Allerdings unterscheidet sich die Planung kabelloser Kommunikationsnetze erheblich von der Planung kabelgebundener Anlagen. Die Funkwellenausbreitung im Gebäude muß bei der Installation der Sendeanlagen zusätzlich berücksichtigt werden. Mit computerunterstützter Planung gelangt man schnell zu konkreten Aussagen, die in der Qualität mit denen eines Experten vergleichbar sind. Das ist die wichtigste Erkenntnis aus der Entwicklung des Softwareprototyps POPULAR (Planning of Picocellular Radio) [9]. Das Planungssystem war zum Zeitpunkt seiner Entwicklung 1995 eines der ersten Systeme mit dieser Funktionalität und wurde 1996 von einer amerikanischen Fachpublikation [9] als eine der weltweit innovativsten Anwendungen im Bereich Telekommunikation ausgewählt.

POPULAR wurde innerhalb eines Mannjahres am ECRC als voll funktionsfähiger Demonstrator implementiert und mit gleichem Zeitaufwand von einem Diplomanten des Instituts für Kommunikationsnetze an der Technischen

[4] Am ECRC in Zusammenarbeit mit der LMU.

Universität Aachen bei Siemens, Abteilung Forschung und Entwicklung, München, zum Prototyp weiterentwickelt. Das Programm für POPULAR ist nur 4000 Zeilen lang und vollständig in einer Constraintlogikprogrammiersprache geschrieben, inklusive der Grafik für die Benutzerschnittstelle, die etwa die Hälfte des Programms braucht.

POPULAR behandelt einerseits ein typisches kombinatorisches Suchproblem, anderseits ist es aber ungewöhnlich wegen der geometrischen Constraints, die zu optimieren sind. Zur Planung simuliert dieses Werkzeug die Funkwellenausbreitung in Gebäuden und optimiert die Anzahl und die Plazierung von Sendeanlagen für lokale, kabellose Kommunikationsnetze.

Die Funkausbreitung wird mithilfe von Testpunkten simuliert. Jeder dieser Testpunkte, die entlang eines drei-dimensionalen Rasters im Gebäude plaziert werden, stellt einen potentiellen Empfänger dar. Für jeden Testpunkt wird die sogenannte „Funkzelle" berechnet. Das ist der Bereich, in dem ein Sender liegen muß, damit der Empfänger mit einem ausreichend guten Signal versorgt werden kann. Dazu wird eine fiktive Funkwelle in einer ausreichenden Anzahl von Richtungen untersucht. Der Ausbreitungspfad wird durch das gesamte Gebäude hindurch bis zur minimal zum Empfang notwendigen Feldstärke verfolgt und dabei die Dämpfung des Signales durch Wände und Decken berücksichtigt.

In der Optimierungsphase wird zuerst für jede Funkzelle folgendes Constraint aufgestellt: Es muß mindestens einen Sender in der Funkzelle geben, damit der zugehörige Testpunkt abgedeckt werden kann. Nun versucht man Senderpositionen zu finden, die möglichst viele Funkzellen gleichzeitig abdecken können. Geometrisch gesprochen muß ein solcher Sender dann im Schnitt der Funkzellen liegen, die er versorgt. Auf diese Weise wird eine erste Lösung berechnet. Selbst wenn nun jeder Sender möglichst viele Funkzellen abdeckt, muß dies in der Gesamtheit nicht zu einer optimalen Lösung mit einer minimalen Anzahl von Sendern führen. Daher wird nun mithilfe eines speziellen Suchverfahrens die Anzahl der Sender verkleinert. Dabei versucht man wiederholt, eine Lösung mit einer kleineren Senderanzahl als bei der letzten Lösung zu finden. Kann man keine Lösung mehr finden, so bietet die letzte Lösung die optimale, d. h. minimale Senderanzahl.

5 Zusammenfassung und Ausblick

Noch vor wenigen Jahren wurden constraintbasierte Systeme der Forschung zugezählt, heute sind sie Stand der Technik und haben sich im kommerziellen Praxiseinsatz bewährt. Erfolgreiche Anwendungen existieren in der Produktions- und Ressourcenplanung, Personalplanung, Transportoptimierung, Layoutgenerierung und in CAD-Systemen. Diese praktischen Erfahrungen werfen wiederum neue wissenschaftliche Fragen auf, z. B. nach flexibleren und beweisbar korrekten Constraintlösern. Einen Einblick in aktuelle Forschungsergebnisse und Anwendungen bieten auch die Referenzen [1, 11, 7, 16, 2, 4].

Mit Constrainttechnologie lassen sich nicht nur kombinatorische Suchprobleme effizienter und flexibler lösen. Unserer Einschätzung nach können damit allgemein berechnungsorientierte Anwendungen, sei es in der Finanzberatung, Stundenplanung oder in der Wettervorhersage, um die Behandlung von ungenauen Angaben erweitert werden. Wir erwarten in den nächsten Jahren ein weiteres rasantes Wachstum für diese Technologie, vor allem auch in Deutschland.

Aus Platzmangel haben wir die Klasse der nebenläufigen constraintbasierten Programmiersprachen [13] nicht erwähnt, in denen Prozesse miteinander durch Abfragen und Einfügen von Constraints interagieren. Constraints finden zudem nicht nur in Programmiersprachen verstärkt Anwendung, sondern auch in Datenbanken [6], wo Constraints es ermöglichen, viele (u. U. unendlich viele) Datenbankeinträge zu einem Eintrag zusammenzufassen. Dies ist vor allem bei der Speicherung von zeitlicher und räumlicher Information von Nutzen.

References

1. F. Benhamou and A. Colmerauer, editors. *Constraint Logic Programming: Selected Research*. MIT Press, Cambridge, MA, USA, 1993.
2. E. C. Freuder, editor. *Second International Conference on Principles and Practice of Constraint Programming CP'96*. Springer LNCS 1118, August 1996.
3. T. Frühwirth. Constraint handling rules. In A. Podelski, editor, *Constraint Programming: Basics and Trends*, LNCS 910. Springer-Verlag, March 1995.
4. T. Frühwirth and S. Abdennadher. *Constraint-Programmierung*. Springer Verlag, Heidelberg, September 1997.
5. T. Frühwirth and S. Abdennadher. Der Mietspiegel im Internet: Ein Fall für Constraint-Logikprogrammierung. *KI 1/97, Themenheft Constraints*, April 1997.
6. D. Q. Goldin and P. C. Kanellakis. Constraint query algebras. *Constraints Journal*, 1(1+2):45–83, September 1996.
7. P. Van Hentenryck and V.J. Saraswat. *Principles and Practice of Constraint Programming*. MIT Press, Cambridge, MA, USA, April 1995.
8. M. Höhfeld and G. Smolka. Definite relations over constraint languages. LILOG Report 53, IWBS, IBM, Stuttgart, Germany, October 1988.
9. P. Brisset J.-R. Molwitz and T. Frühwirth. Planning cordless business communication systems. In *IEEE Expert Magazine, Special Track on Intelligent Telecommunications*, January 1996.
10. J. Jaffar and J. L. Lassez. Constraint logic programming. In *Proceedings of the* 14^{th} *ACM Symposium on Principles of Programming Languages POPL-87, Munich, Germany*, pages 111–119, 1987.
11. J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 20:503–581, 1994.
12. M. J. Maher. Logic semantics for a class of committed-choice programs. In J.-L. Lassez, editor, *Proceedings of the Fourth International Conference on Logic Programming*, pages 858–876. MIT Press, May 1987.
13. V.A. Saraswat. *Concurrent Constraint Programming*. MIT Press, 1993.
14. L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, 1994.
15. P. van Hentenryck, H. Simonis, and M. Dincbas. Constraint satisfaction using constraint logic programming. *Artificial Intelligence*, 58(1-3):113–159, 1992.
16. M. Wallace. Practical applications of constraint programming. *Constraints Journal*, 1(1,2):139–168, September 1996. Kluwer Academic Publishers.

This article was processed using the LaTeX macro package with LLNCS style