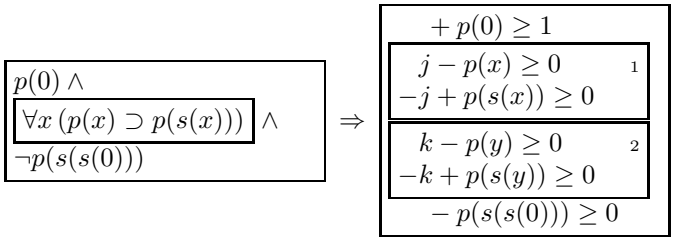
Tableaubasiertes prädikatenlogisches Beweisen mit Mixed integer programming

Bernhard Beckert, Reiner Hähnle, Klaus Ries

Universität Karlsruhe
Institut für Logik, Komplexität und Deduktionssysteme
Am Fasanengarten 5, 76128 Karlsruhe
{beckert,haehnle,ries}@ira.uka.de

In letzter Zeit werden verschiedene Anätze zum Automatischen Beweisen mit Hilfe von Integer programming[1] (IP) entwickelt (2; 3; 1; 5), die als Weiterführung des Ansatzes von Lee & Plaisted (4) gesehen werden können. Hier soll ein auf dem Tableaukalkül basierendes Verfahren für die volle Prädikatenlogik vorgestellt werden. Es ist eine Erweiterung der in (1) beschriebenen Übersetzung aussagenlogischer Formeln in Integer-programming-Probleme.

Eine *prädikatenlogische* Formelmenge wird in ein IP-Problem übersetzt, das *rigide* (d. h. nicht implizit universell quantifizierte) freie Variablen enthält (das nebenstehende Diagramm zeigt ein Beispiel). Von jedem einer universell quantifizierten Teilformel entsprechenden Teil des IP, wird eine bestimmte Anzahl n von Kopien mit jeweils neuen Variablen erzeugt (im Beispiel ist $n = 2$ für die einzige vorhandene universelle Formel). Die Ausgangsformel ist nun genau dann unerfüllbar, wenn es für jede universell quantifizierte Teilformel eine Anzahl n von Kopien gibt und eine (Grund-)Substitution σ der freien Variablen des IP, so daß das instantiierte IP unlösbar ist (im Beispiel kann man die Substitutionen $\{x/s(0), y/0\}$ oder $\{x/0, y/s(0)\}$ verwenden).



Die Variablen j und k sind im ursprünglichen Problem nicht vorhanden. Sie entsprechen grob den Definitionen (manchmal auch „Abkürzungen“ oder „*renamings*“ genannt), die in polynomiellen Algorithmen zur Berechnung einer KNF verwendet werden. Gleichzeitig repräsentieren diese Variablen jedoch implizit die Verzweigungen eines partiell expandierten semantischen Tableaus für die Eingabeformelmenge. Dies bedeutet, daß bei geschickter Wahl der Verzweigungsvariablen die Tableaustruktur im wesentlichen im resultierenden IP mitrepräsentiert ist. Es stellt sich heraus, daß es günstig sein kann, Verzweigungs*prädikate* statt Verzweigungs*variablen* einzuführen, d. h. j, k usw. mit einem Teil der freien Variablen zu parametrisieren.

Die erforderliche Anzahl von Kopien wird durch inkrementelle Approximation ermittelt,

die möglichen Substitutionen durch eine Analyse der Link-(Konnektions-)Struktur. Da diese Analyse global erfolgt — also nicht auf Zweigbasis, wie es beim Tableaukalkül der Fall ist, — ist wesentlich weniger Backtracking als bei einem Tableauverfahren nötig.

Eine Implementierung des vorgestellten Verfahrens ist nahezu fertiggestellt. Den Kern bildet die Komponente *lp++* zur Lösung der entstehenden MIP-Probleme. Sie ist in C++ implementiert und über eine an Effizienz orientierte Schnittstelle in Prolog integriert. *lp++* beruht auf einem dualen Simplexverfahren auf der Basis der Matrixdarstellung nach Beale zur Lösung des linearen Programmierproblems, bei dem im Gegensatz zu Mixed integer programming nicht die Ganzzahligkeit von Variablen vorgeschrieben werden kann. Darauf aufbauend werden MIP-Probleme durch ein Branch-and-bound-Verfahren gelöst. Mit einigen am Davis-Putnam-Algorithmus orientierten Optimierungen wird mit *lp++* eine hohe Geschwindigkeit bei aussagenlogischen Testproblemen erreicht. Als wichtige Erweiterung für das prädikatenlogische Beweisen wurde *lp++* um Methoden zur inkrementellen Lösung einer Serie von MIP-Problemen und einer expliziten Modellierung von „Links“ erweitert.

Neben den besonderen Möglichkeiten, die die Verwendung von MIP bietet, wie beispielsweise IP-spezifische Präprozessoren einzusetzen,[2] die Berechnung von Links (Konnektionen) als Optimierungsproblem zu formulieren, und Metawissen durch lineare Ungleichungen darzustellen, können auch die Vorteile des Tableaukalküls ausgenutzt werden. So wird keinerlei Normalform vorausgesetzt, es ist möglich, nichtklassische Logiken, insbesondere mehrwertige, zu behandeln, Gleichheitstheorien können — syntaktisch uneingeschränkt — effizient eingebaut werden, und eine optimierte Skolemisierung sowie die Erzeugung einer definitorischen Normalform fallen gleichsam als Nebenprodukt ab. Da das aus der Übersetzung nach jedem inkrementellen Schritt resultierende Ergebnis einer partiell instantiierten Menge von Klauseln entspricht, können auch resolutionstypische *Forward-reasoning*-Schritte mit der inkrementellen Übersetzung verzahnt werden.

[1] R. Hähnle. A new translation from deduction into integer programming. In J. Calmet and J. A. Campbell, editors, *Proc. Int. Conf. on Artificial Intelligence and Symbolic Mathematical Computing AISMC-1, Karlsruhe, Germany*, pages 262–275. Springer, LNCS 737, 1992.

[2] J. N. Hooker. New methods for computing inferences in first order logic. Working Paper, GSIA, CMU Pittsburgh, April 1992.

[3] V. Kagan, A. Nerode, and V. Subrahmanian. Computing definite logic programs by partial instantiation and linear programming, 1993.

[4] S.-J. Lee and D. A. Plaisted. Eliminating duplication with the hyper-linking strategy. *Journal of Automated Reasoning*, 9(1):25–42, 1992.

[5] K. Ries and R. Hähnle. Prädikatenlogisches Beweisen mit gemischt ganzzahliger Optimierung. Ein tableaubasierter Ansatz. In *Working Notes of Workshop Künstliche Intelligenz und Operations Research, Berlin (published as Tech Report, MPI für Informatik, Saarbrücken)*, 1993.

[1] Ein (Mixed-)Integer-programming-Problem ist ein Problem der linearen Programmierung, d. h. eine Menge linearer Ungleichungen (Constraints), mit der zusätzlichen Einschränkung, daß (einige der) Variablen ganzzahlig sein müssen. Eine Lösung des Problems ist eine Variablenbelegung, die die Constraints erfüllt und zugleich eine lineare Kostenfunktion minimiert.

[2] Der Präprozessor des kommerziellen Werkzeuges CPLEX löst beispielsweise eine Standardformulierung der Pigeonhole-Probleme in linearer Zeit, da die Rekursionsstruktur erkannt wird.