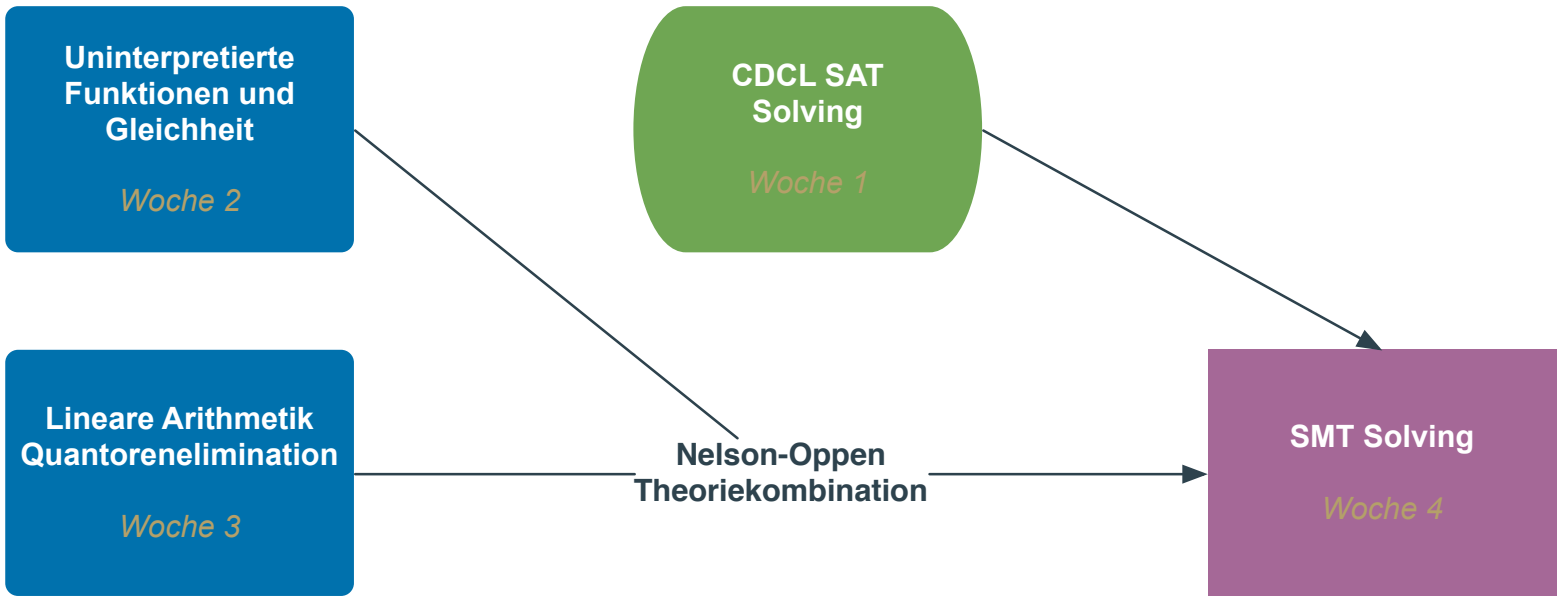
Automatisches Beweisen—Vertiefung
Entscheidbare FOL Fragmente
Lineare Arithmetik & Quantorenelimination

Christoph Zengler

Arbeitsbereich Symbolisches Rechnen
Prof. Dr. Wolfgang Küchlin
Universität Tübingen

11. Dezember 2012

Wir erinnern uns...



Lineare Arithmetik

Syntax

- **Linear:** Keine Multiplikation zwischen Variablen (d.h. auch keine Potenzen)
- $5x$ ist Abkürzung für $x + x + x + x + x$

EBNF-Grammatik für Lineare Arithmetik

$$
\begin{array}{lcll}
\text{Summe} & = & \text{Term} \mid \text{Summe} + \text{Term} & \\
\text{Term} & = & \textit{Variable} \mid \textit{Konstante} \mid \textit{Konstante} \cdot \textit{Variable} & \\
 & & & \\
\text{Formel} & = & \top \mid \bot & \text{Konstanten} \\
 & \mid & \text{Summe} = \text{Summe} & \\
 & \mid & \text{Summe} \leq \text{Summe} & \\
 & \mid & \text{Summe} < \text{Summe} & \\
 & \mid & \text{Formel} \wedge \text{Formel} & \text{Konjunktion}
\end{array}
$$

- Wir betrachten nur Konjunktionen
- $a - b$ ist *syntactic sugar* für $a + -b$
- $\leq$ und $<$ können durch $\geq$ und $>$ ersetzt werden (durch Negation der Koeffizienten)

Lineare Arithmetik

Semantik

- Universum ist $\mathbb{Q}$ oder $\mathbb{Z}$
- Arithmetische Symbole $+, -, \cdot$ werden wie üblich über $\mathbb{Q}$ und $\mathbb{Z}$ interpretiert
- Konstanten werden als 0-stellige Funktionen mit dem jeweiligen Wert in $\mathbb{Q}$ oder $\mathbb{Z}$ interpretiert
- Prädikate $=, \leq, \geq$ werden wie üblich über $\mathbb{Q}$ und $\mathbb{Z}$ interpretiert

Damit ist eine Interpretation $\mathcal{M}_{\mathrm{LA}}$ fixiert.

- Belegung β bildet Variablen nach $\mathbb{Q}$ oder $\mathbb{Z}$ ab
- Im Gegensatz zur allgemeinen FOL reicht für die Erfüllbarkeit die **Existenz** einer Belegung β aus, so dass die Formel zu `true` evaluiert.

Definition (Erfüllbarkeit in der Linearen Arithmetik)

Eine Formel φ in LA ist erfüllbar gdw. es eine Belegung β gibt, so dass $\texttt{holds}(\mathcal{M}_{\mathrm{LA}}, \beta, \varphi) = \texttt{true}$ gilt.

$\mathbb{Z}$ vs. $\mathbb{Q}$

Wir betrachten $\mathbb{Z}$ und $\mathbb{Q}$ als Domänen

? Was wollen wir wissen?

Ist eine Formel in Linearer Arithmetik erfüllbar, wenn ja mit welcher Belegung der Variablen.

Beispiel (Formel in Linearer Arithmetik)

$$3x_1 + 2x_2 \leq 5x_3 \wedge 2x_1 - 2x_2 = 0$$

- Für $\mathbb{Q}$ ist das Problem in Polynomialzeit entscheidbar
- Für $\mathbb{Z}$ ist das Problem NP-vollständig (Pseudo Boolesche Optimierung)
- Nimmt man in die Sprache noch die “echte” Multiplikation hinzu, so ist das Problem für $\mathbb{Z}$ unentscheidbar (Gödelscher Unvollständigkeitssatz)

Anwendungsbeispiel

Code Optimierung im Compiler

Beispiel (C Programm)

```
for(i=1; i<=10; i++)
  a[j+i]=a[j]
```

- `a[j]` wird 10 mal ausgelesen
- Zugriff auf Speicher ist sehr langsam im Vergleich zu Werten in Registern
- **Frage:** Kann man `a[j]` nur einmal am Anfang der Schleife auslesen?
- **Antwort:** Nur wenn `a[j]` den Wert in der Schleife nicht ändert, also nie auf `a[j]` geschrieben wird

$$i \geq 1 \wedge i \leq 10 \wedge j + i = j$$

ist nicht erfüllbar, daher ist die Optimierung sicher.

Übersicht

Simplex

- Einer der ältesten Algorithmen (Danzig, 1947)
- Finde einen optimalen Wert für eine Zielfunktion bei gegebenen Constraints über reellen Zahlen
- Constraints + Zielfunktion = **Lineares Programm**
- Entscheidungsproblem **Allgemeiner Simplex**: Ist eine gegebene Menge von linearen Constraints über $\mathbb{Q}$ oder $\mathbb{R}$ erfüllbar oder nicht

Integer Linear Programming (ILP)

- Selbe Fragestellung über $\mathbb{Z}$
- Algorithmus: **Branch and Bound**

Zwei einfache Ansätze

- **Fourier Motzkin** Variablen Elimination für $\mathbb{R}$ oder $\mathbb{Q}$
- **Omega Test** für $\mathbb{Z}$

Vorverarbeitung

Lineare Systeme über $\mathbb{Q}$, $\mathbb{R}$ oder $\mathbb{Z}$

- unabhängig vom gewählten Entscheidungsverfahren

Erste Vereinfachung

$$x_1 + x_2 \leq 2, x_1 \leq 1, x_2 \leq 1$$

Die erste Ungleichung ist überflüssig

Allgemeine Formulierung

Für eine Menge

$$S = \left\{ a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b, l_j \leq x_j \leq u_j \text{ für } j = 0, \ldots, n \right\}$$

von Constraints ist das erste Constraint redundant, wenn

$$\sum_{j|a_j>0} a_j u_j + \sum_{j|a_j<0} a_j l_j \leq b.$$

Vorverarbeitung

Lineare Systeme über $\mathbb{Q}$, $\mathbb{R}$ oder $\mathbb{Z}$

Zweite Vereinfachung

$$2x_1 + x_2 \leq 2, x_2 \geq 4, x_1 \leq 3$$

Aus dem ersten und zweiten Constraint folgt $x \leq -1$, d.h. das dritte Constraint kann spezialisiert werden.

Allgemeine Formulierung

Für $a_0 > 0$ gilt

$$x_0 \leq \left(b - \sum_{j|j>0, a_j>0} a_j l_j - \sum_{j|a_j<0} a_j u_j \right) / a_0$$

und für $a_0 < 0$ gilt

$$x_0 \geq \left(b - \sum_{j|a_j>0} a_j l_j - \sum_{j|j>0, a_j<0} a_j u_j \right) / a_0.$$

Vorverarbeitung

Lineare Systeme über $\mathbb{Z}$

Strikte Ungleichungen

$$\sum_{1 \leq j \leq n} a_i x_i < b$$

können in nicht-strikte Ungleichungen

$$\sum_{1 \leq j \leq n} a_i x_i \leq b - 1$$

umgewandelt werden.

Hinweis!

Systeme in $\mathbb{Q}$ können durch Multiplikation mit dem kgV der Nenner in Systeme mit rein ganzzahligen Koeffizienten überführt werden.

Fourier-Motzkin Variablenelimination

- FM bekommt als Eingabe eine Konjunktion von linearen Constraints über reellen Variablen
- m Constraints und $x_1, \ldots, x_n$ Variablen

Auflösen von Gleichungen

Wir betrachten Constraints der Form

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i$$

1. Wähle ein x_j mit Koeffizientem $a_{ij} \neq 0$. O.B.d.A. sei dies x_n.
2. Löse die Gleichung nach x_n auf

$$x_n = \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j$$

3. Substituiere die rechte Seite für x_n im gesamten System S und entferne den Constraint i

Auflösen von Gleichungen

Beispiel (Auflösen von Gleichungen)

Betrachten wir die Gleichung

$$2x_1 + 3x_2 - 4x_3 = 8$$

Wir lösen die Gleichung nach x_3 auf

$$x_3 = 2 - \frac{1}{2}x_1 - \frac{3}{4}x_2$$

und substituieren nun überall die rechte Seite für x_3 (d.h. x_3 wird aus dem System eliminiert).

Was bleibt ist ein System von m Ungleichungen der Form

$$\bigwedge_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_j \leq b_i$$

Auflösen von Ungleichungen

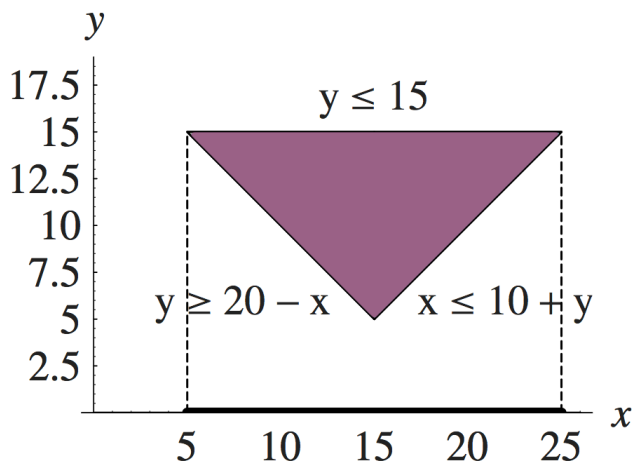
Idee!

Wähle eine Variable aus und projiziere ihre Constraints auf den Rest des Systems und eliminiere so diese Variable.

Beispiel (Auflösen einer Ungleichung)

- Die Constraints $x \leq y + 10,\ y \leq 15,\ y \geq -x + 20$ bilden einen Dreieck im $\mathbb{R}^2$
- Projektion auf die x-Achse ergibt eine Linie gegeben durch die Constraints

$$5 \leq x \leq 25$$



Auflösen von Ungleichungen

- Pro Projektionsschritt wird das Problem um eine Variable kleiner
- Aber möglicherweise mehr Constraints
- Iteriere bis nur noch eine Variable übrig bleibt
- Variablenordnung ist entscheidend (Heuristiken)

Umschreiben der Constraints

Wir eliminieren wieder x_n und betrachten das i-te Constraint

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i.$$

- O.B.d.A. betrachten wir den Fall $a_{in} > 0$. Dann kann das Constraint umgeschrieben werden zu

$$a_{in} x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij} x_j.$$

Umschreiben der Constraints

$$a_{in} x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij} x_j .$$

- Ist $a_{in} = 0$ braucht das Constraint nicht weiters beachtet werden
- Ansonsten teilen wir durch a_{in}:

$$x_n \leq \underbrace{\frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j}_{\beta_i} .$$

(Für $a_{in} > 0$ ist das neue Constraint eine Obergrenze, für $a_{in} < 0$ eine Untergrenze)

- Die rechte Seite des neuen Constraints bezeichnen wir mit β_i.

Idee!

Um eine Variable x_n zu eliminieren, löse alle Ungleichungen, die sie enthalten, nach x_n auf und bilde alle möglichen Paare von Ober- und Untergrenzen auf ihr.

Eliminieren von Variablen

Ungebundene Variablen

- Hat eine Variable nach Auflösung in allen Constraints **nur** Unter- oder nur Obergrenzen, können sie und alle Constraints, die sie enthalten einfach entfernt werden.
- Dies kann andere Variablen ungebunden machen, also iteriere diesen Schritt

Gebundene Variablen

- Zähle alle möglichen Paare aus Untergrenzen β_l und Obergrenzen β_u auf
- Für jedes Paar gilt dann: $\beta_l \leq x_n \leq \beta_u$
- Füge das neue Constraint $\beta_l \leq \beta_u$ hinzu
- Entsteht dadurch ein Widerspruch, ist das System nicht erfüllbar

Beispiel für Fourier-Motzkin

$$
\begin{array}{rrrrcr}
x_1 & -x_2 & & & \leq & 0 \\
x_1 & & -x_3 & & \leq & 0 \\
-x_1 & +x_2 & +2x_3 & & \leq & 0 \\
 & & -x_3 & & \leq & -1 \\
 & & & x_4 & \leq & 8
\end{array}
$$

Beispiel (Fourier-Motzkin)

1. x_4 ist ungebunden (nur Obergrenze), d.h. x_4 und Constraint 5 können entfernt werden
2. Elimination von x_1:
   1. Umschreiben der Constraints 1-3:

$$
\begin{array}{rcl}
x_1 & \leq & x_2 \\
x_1 & \leq & x_3 \\
x_2 + 2x_3 & \leq & x_1
\end{array}
$$

   2. Zwei Obergrenzen und eine Untergrenze $\Rightarrow$ Zwei neue Constraints $2x_3 \leq 0$ und $x_2 + x_3 \leq 0$

Beispiel für Fourier-Motzkin

Neues Problem nach Elimination von x_4 und x_1

$$\begin{array}{rrcr} & 2x_3 & \leq & 0 \\ x_2 & +x_3 & \leq & 0 \\ & -x_3 & \leq & -1 \end{array}$$

Beispiel (Fourier-Motzkin)

1. x_2 ist nun ungebunden (nur Obergrenze), d.h. x_2 und Constraint 2 können entfernt werden
2. Es bleibt das System

$$\begin{array}{rcr} 2x_3 & \leq & 0 \\ -x_3 & \leq & -1 \end{array}$$

Kombination des Unter- und Obergrenze von x_3 ergibt $1 \leq 0$

$\Rightarrow$ Ein Widerspruch, also ist das System nicht erfüllbar

Komplexität von Fourier-Motzkin

- In jedem Schritt kann sich die Anzahl der Constraints im worst-case von m auf $m^2/4$ erhöhen
- Insgesamt also $m^{2^n}/4^n$ Constraints, d.h. doppelt-exponentiell

$\Rightarrow$ Nur für kleine Systeme geeignet

Das Branch & Bound Verfahren

- Verbreitetes Verfahren zum Lösen von ganzzahligen linearen Programmen (ILP)
- Wir betrachten die Entscheidungsvariante
- Eingabe: Nicht-strikte lineare Constraints
- Alle Variablen müssen mit Werten aus $\mathbb{Z}$ belegt werden
- Strikte Constraints können durch Vorverarbeitung nicht-strikt gemacht werden

Definition (Relaxiertes Problem)

Zu einem gegebenen ganzzahligen linearen System S ist das relaxierte Problem $\texttt{relaxed}(S)$ das System S ohne die Bedingung, dass die Variablen aus $\mathbb{Z}$ belegt werden.

Annahme:

- Es gibt ein Verfahren $\text{LP}_{\text{feasible}}$, dass für ein lineares Programm S entscheidet, ob es erfüllbar ist (dann wird eine Belegung ausgegeben) oder unerfüllbar ist $\Rightarrow$ Z.B. Fourier-Motzkin

Der Branch & Bound Algorithmus

Algorithmus: branchAndBound(S)

Eingabe: Ein ganzzahliges lineares System S
Ausgabe: SAT wenn S erfüllbar ist, UNSAT sonst

---

searchIntegralSolution(S)
return UNSAT

Prozedur searchIntegralSolution(S):

$res = \text{LP}_{\text{feasible}}(\text{relaxed}(S))$
if res = UNSAT **then**
  return
else
  if res *is integral* **then**
    abort SAT
  else
    Select a variable v that is assigned a nonintegral value r
    searchIntegralSolution($S \cup (v \leq \lfloor r \rfloor)$)
    searchIntegralSolution($S \cup (v \geq \lceil r \rceil)$)

Der Branch & Bound Algorithmus

Illustration an einem Beispiel

Beispiel (Branch & Bound Algorithmus)

- Variablen von S: $x_1, \ldots, x_4$
- Angenommen Lösung von $\mathtt{LP}_{\text{feasible}} = (1, 0.7, 2.5, 3)$

1. Wähle Variable x_2
2. Versuche das System $S \cup \{x_2 \leq 0\}$ zu lösen
   - Wird keine Lösung gefunden kommt dieser Branch zurück
3. Versuche das System $S \cup \{x_2 \geq 1\}$ zu lösen
   - Wird keine Lösung gefunden, gibt es keine Lösung, `searchIntegralSolution` terminiert und `branchAndBound` gibt `UNSAT` zurück
4. Wird eine Lösung gefunden wird das Verfahren iteriert

Vollständigkeit

⚠ Achtung!

Das Verfahren ist so nicht vollständig!

$$1 \leq 3x - 3y \leq 2$$

hat z.B. keine ganzzahligen Lösungen, jedoch unendlich viele reelle.

Definition (Small-Model-Property)

Die Small-Model-Property besagt, dass jede erfüllbare Formel ein Modell mit endlichen Grenzen hat.

- Die Formeln, die wir betrachten, haben diese Small-Model-Property
- Die Grenzen sind berechenbar
- Wir können das Verfahren abbrechen, sobald wir über die berechneten Grenzen einer Variable hinausgehen

Quantorenelimination

Definition (Quantoreneliminationsverfahren)

Ein Quantoreneliminationsverfahren für ein Modell $\mathcal{M}$ ist ein Algorithmus, der eine Formel φ in eine quantorenfreie Formel φ' transformiert, so dass $\vDash_{\mathcal{M}} \varphi \leftrightarrow \varphi'$ gilt.

- Nicht alle Modelle erlauben QE
- Sehr feine Unterschiede:
  - $(\mathbb{R}, 0, 1, +, -, \cdot)$ erlaubt keine QE, $(\mathbb{R}, 0, 1, +, -, \cdot, <)$ erlaubt QE.
  - $(\mathbb{Z}, 0, 1, +, -, <)$ erlaubt keine QE, $(\mathbb{Z}, 0, 1, +, -, <, \{\equiv_n\}_{n \in \mathbb{N}})$ erlaubt QE (Presburger Arithmetik).
- Quantorenfreie Formel ist nicht nur erfüllbarkeitsäquivalent, sondern äquivalent
- Eigenes großes Forschungsgebiet (z.B. Collins, Weispfenning, Sturm, . . .)

Einige Beispiele für QE

Beispiel (Reelle Zahlen $\mathbb{R}$)

$$\varphi = \exists x[ax + b = 0]$$

ist äquivalent zu

$$\varphi' = b = 0 \vee a \neq 0$$

$$\varphi = \forall x[ax + b = 0]$$

ist äquivalent zu

$$\varphi' = a = 0 \wedge b = 0$$

Beispiel (Presburger Arithmetik $\mathbb{Z}$)

$$\varphi = \exists x[2x = a \wedge b < x \wedge x < c]$$

ist äquivalent zu

$$\varphi' = a \equiv_2 0 \wedge 2b < a < 2c$$

Quantoren in der Aussagenlogik—1

⚠ AL vs. FOL

In der Aussagenlogik gibt es keine Quantoren.

Wir müssen uns einen Weg überlegen, AL in FOL einzubetten.
Zwei Möglichkeiten sind denkbar:

Alternative 1: Prädikate

Betrachte aussagenlogische Variablen als 0-stellige Prädikate:

Beispiel (AL Variablen als Prädikate)

$$(a \wedge b \vee c) \vee \neg(a \vee \neg c)$$

wird zu

$$(A() \wedge B() \vee C()) \vee \neg(A() \vee \neg C())$$

Problem: Quantifizierung über Prädikate ist nicht mehr FOL!

Quantoren in der Aussagenlogik—2

Alternative 2: Neue Sprache

Betrachte die Sprache: $(\&^{(2)}, |^{(2)}, !^{(1)}, 0^{(0)}, 1^{(0)}, =^{(2)})$ wobei $\&, |, !, 0, 1$ Terme sind und $=$ Prädikat.

Beispiel (AL Variablen als Atomare Formeln)

$$(a \wedge b \vee c) \vee \neg(a \vee \neg c)$$

wird zu

$$((a \,\&\, b \mid c) \mid !(a \mid !c)) = 1$$

oder aber auch:

$$((a = 1) \wedge (b = 1) \vee (c = 1)) \vee \neg((a = 1) \vee (c = 0))$$

Wir bevorzugen diesen Ansatz!

Quantifizierte Boolesche Formeln

Wir haben ab jetzt *Alternative 2* im Hinterkopf und schreiben der Einfachheit halber: $\exists x \forall y [(x \wedge y) \vee (y \wedge \neg x)]$

4 Arten von quantifizierten Booleschen Formeln

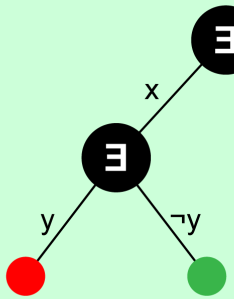
1. *Rein existentiell quantifizierte Sätze:* Reines SAT Problem (existiert eine Belegung?)
2. *Rein allquantifizierte Sätze:* Tautologietest (gilt eine Formel für alle Belegungen)
3. *Beliebig quantifizierte Sätze:* Sog. QBF Problem (Quantified Boolean Formulas)
4. *Beliebig quantifizierte Formeln:* Volle Boolesche QE

Anmerkung

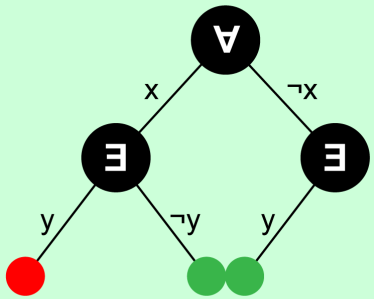
1–3 sind gut erforschte Gebiete. 4 ist aktuelle Forschung am Arbeitsbereich (BSc/MSc Arbeiten!).

QBF

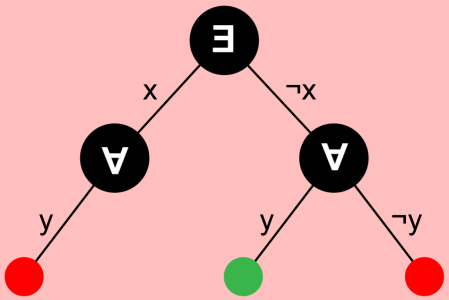
$\exists x\ \exists y\ ((x \vee y) \wedge (\neg x \vee \neg y))$



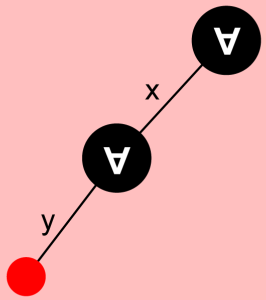
$\forall x\ \exists y\ ((x \vee y) \wedge (\neg x \vee \neg y))$



$\exists x\ \forall y\ ((x \vee y) \wedge (\neg x \vee \neg y))$



$\forall x\ \forall y\ ((x \vee y) \wedge (\neg x \vee \neg y))$



Lösung mit einer Variante des CDCL Algorithmus:

- Backtracking auch im SAT Fall
- Näheres siehe SAT Vorlesung

Echte Boolesche QE

Ein naiver Ansatz: *Substitute & Simplify* (Seidl & Sturm 2003)

Beobachtungen

- Wertebereich der Variablen ist immer von Kardinalität 2

$\Rightarrow$ All- und Existenzquantor können explizit ausgeschrieben werden

$$\forall x \varphi \equiv \mathtt{subst}(\{x \mapsto 1\}, \varphi) \wedge \mathtt{subst}(\{x \mapsto 0\}, \varphi)$$

$$\exists x \varphi \equiv \mathtt{subst}(\{x \mapsto 1\}, \varphi) \vee \mathtt{subst}(\{x \mapsto 0\}, \varphi)$$

Algorithmus: $ssss(\varphi)$

Eingabe: Eine beliebig quantifizierte aussagenlogische Formel
Ausgabe: quantorenfreies φ' mit $\varphi' \equiv \varphi$

1. Konvertiere φ in PNF
2. Eliminiere Quantoren von innen nach außen nach obiger Regel
3. Simplifiziere die Formel nach jeder Elimination

Beispiel für Substitute & Simplify

Beispiel (Substitute & Simplify)

$$\varphi = \exists x \forall y (x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w))$$

Substituiere y:

$$\exists x [(x \vee \top \vee \neg u) \wedge (\neg x \vee \neg \top \vee w) \wedge (x \vee \bot \vee \neg u) \wedge (\neg x \vee \neg \bot \vee w)]$$

Simplifiziere:

$$\exists x [(\neg x \vee w) \wedge (x \vee \neg u)]$$

Substituiere x:

$$((\neg \top \vee w) \wedge (\top \vee \neg u)) \vee ((\neg \bot \vee w) \wedge (\bot \vee \neg u))$$

Simplifiziere:

$$w \vee \neg u$$

Fortgeschrittene Lösungsansätze

- Top-Level DPLL für die freien Variablen, der an jedem Blatt wiederum SAT/QBF aufruft (1)
- Resolutionsbasierter Ansatz (Clause Distribution) (2)
- SAT-basierter Ansatz (Model Counting) (2)
- Knowledge Compilation Ansatz (DNNF) (2)

Aktuelle Forschung am Arbeitsbereich

(1) *Thomas Sturm, Christoph Zengler*: **Parametric Quantified SAT Solving**, ISSAC 2010

(2) *Andreas Kübler, Wolfgang Küchlin, Christoph Zengler*: **New Approaches to Boolean Quantifier Elimination**, ACM Communications in Computer Algebra 45

QE in der Linearen Arithmetik

- Mit Fourier-Motzkin kennen wir bereits ein Verfahren, um existentielle Quantoren zu eliminieren
- Nutze Dualität von $\exists x$ und $\forall x$ aus, um Allquantoren zu eliminieren

Beispiel (QE mit Fourier-Motzkin)

$$\forall x \exists y \exists z [y + 1 \leq x \wedge z + 1 \leq y \wedge 2x + 1 \leq z]$$

Elimination von z:

$$\forall x \exists y [y + 1 \leq x \wedge 2x + 1 \leq y - 1]$$

Elimination von y:

$$\forall x [2x + 2 \leq x - 1]$$

Dualität von $\forall x$ und $\exists x$:

$$\neg \exists x [2x + 2 > x - 1]$$

äquivalent zu

$$\neg \exists x [x > -3]$$

offensichtlich nicht gültig.

Literaturtipps

Literaturhinweis

- *D. Kroening & O. Strichman.* **Decision Procedures: An Algorithmic Point of View** **Chapter 5 & 9.** Springer, 2008.
- *A. Seidl & T. Sturm* **Boolean Quantification in a First-Order Context.** Proceedings of the CASC 2003.

Web Links

- `http://reduce-algebra.sourceforge.net/` — Computer Algebra System
- `http://redlog.dolzmann.de/` — QE Paket für Redlog