Implementierung eines Deduktionstools über HORN-Klauseln der Prädikatenlogik mit Gleichheit

Verteidigungsvortrag zur Bachelorarbeit

Bearbeiter: Andreas Loth
Betreuer: apl. Prof. Dr.-Ing. habil. Rainer Knauf
Datum: 06.12.2015

TECHNISCHE UNIVERSITÄT
ILMENAU

Gliederung

1. Einleitung
2. Die Paramodulationsregel
3. Das Deduktionstool
4. Implementierung der Paramodulation
5. Demo
6. Fazit und Ausblick

Einleitung

- reale Welt
  - mehrere Namen für ein Individuum
    - Andreas
    - Andy
    - Andy_2639
  - Bezeichnung durch funktionalen Zusammenhang
    - der Nachbar von Frau Schmidt
    - Herr Müller
- Modell der Welt
  - unterschiedliche Bezeichner → unterschiedliche Individuen
    - `andreas` ≠ `andy` ≠ `andy_2639`
    - `nachbar_von(frau_schmidt)` ≠ `herr_mueller`
- Lösung: Modell mit Gleichheit

Die Paramodulationsregel

- Semantik, Anwendung
- Gleichheitsaxiome
- Einschränkungen

Die Paramodulationsregel Semantik, Anwendung

- Regel: $(a \equiv b) \leftarrow p_1 \wedge p_2 \wedge \ldots \wedge p_n$
- Hypothese: $h_1 \wedge h_2 \wedge \ldots \wedge h_m$
  t ist Term in h_i
- $\vartheta_1(a) = \vartheta_2(t)$
  h_i‘: ein Vorkommen von t durch $\vartheta_1(b)$ ersetzen
- neue Hypothese: $\vartheta_2(h_1) \wedge \ldots \wedge \vartheta_2(h_{i-1}) \wedge \vartheta_2(h_i') \wedge \vartheta_1(p_1) \wedge \ldots \wedge \vartheta_1(p_m) \wedge \vartheta_2(h_{i+1}) \wedge \ldots \wedge \vartheta_2(h_n)$

Die Paramodulationsregel Gleichheitsaxiome

- Reflexivität: $(a \equiv a)$
- Symmetrie: $(a \equiv b) \Rightarrow (b \equiv a)$
- Transitivität: $(a \equiv b \wedge b \equiv c) \Rightarrow (a \equiv c)$
- Funktionssubstitutivität:
  $(a_1 \equiv b_1 \wedge \ldots \wedge a_n \equiv b_n)$
  $\Rightarrow (f(a_1, \ldots, a_n) \equiv f(b_1, \ldots, b_n))$
- Prädikatsubstitutivität:
  $(a_1 \equiv b_1 \wedge \ldots \wedge a_n \equiv b_n)$
  $\Rightarrow (p(a_1, \ldots, a_n) \leftrightarrow p(b_1, \ldots, b_n))$

Die Paramodulationsregel Einschränkungen

- geordnete Paramodulation
  - Reduktionsordnung $\succ$
    - strikt, partiell
    - abgeschlossen unter Termersetzungen:
      $a \succ b \Rightarrow \vartheta(a) \succ \vartheta(b)$
    - erfüllt Teiltermeigenschaft:
      b ist echter Teilterm von a $\Rightarrow a \succ b$
  - Ersetzung (a durch b) wird nur ausgeführt, wenn
    - $\vartheta(b) \not\succ \vartheta(a)$ oder
    - $\vartheta(b) \prec \vartheta(a)$

$$(a \equiv b) \leftarrow p_1 \wedge \ldots \wedge p_n$$

Die Paramodulationsregel Einschränkungen

- gerichtete Paramodulation
  - Termersetzungen nur möglich
    - entweder von links-nach-rechts
    - oder von rechts-nach-links

$$(a \equiv b) \leftarrow p_1 \wedge \ldots \wedge p_n$$

Das Deduktionstool

- Arbeitsname: PHP-Prolog

- Arbeitsweise von Prolog
- Syntax
- Plug-ins
- Optimierungen

Das Deduktionstool
Arbeitsweise von Prolog

- Resolution nach Robinson
- Einteilung der Klauseln: Programm, Fragen
- Ordnung der Klauseln, Atomformeln
- Resolution nur zwischen Programm und erster Atomformel des Resolventen
- Tiefensuche mit Backtrack
- Cut-Operator: !

Das Deduktionstool
Syntax

- ähnlich der Syntax von Prolog
- keine Infixnotation
- Zeichensatz: UTF-8
- beliebige Namen durch "…"

Das Deduktionstool
Plug-ins

- Erweiterung des Funktionsumfangs durch
  - Self Evaluating Predicates (SEPs)
  - Rules with own Inference Method (RIMs)
  - Postprozessoren
  - Präprozessoren
  - Prolog-Code

Das Deduktionstool Optimierungen

- Eliminierung von Teilzielen
  - Idempotenz und Kommutativität der Konjunktion → weglassen gleicher Teilziele
  - Prädikate mit Nebeneffekten nicht kommutativ
  - keine Variablenumbenennung beim Vergleich
  - Subsumption auch denkbar

Das Deduktionstool Optimierungen

- Speichern der Historie bereits erreichter Ziele
  - Erkennen von Schleifen im Ableitungspfad
    → nicht terminierende Programme
  - neu abgeleitetes Ziel bereits vorher abgeleitet
    → Backtrack
  - Vergleich ohne Variablenumbenennung, Kommutativität (absolute Gleichheit)
  - zwei Arten: alle Ziele oder nur Ziele auf aktuellem Pfad

Implementierung der Paramodulation

- Syntax
- Einordnung in Tiefensuche mit Backtrack
- Probleme und Lösungen
- Einschränkungen

Impl. der Paramodulation Syntax der Regeln mit "="

- `"="(a, b) :- p1, …, pn.`
  `"="(a, b).`
  - `a` echter Teilterm von `b` oder umgekehrt: geordnet
  - sonst: ungeordnet
- `"="(a, b, d) :- p1, …, pn.`
  `"="(a, b, d).`
  - gerichtet
- `a` und `b`: Terme
  $p_1, \ldots, p_n$: Atomformeln
  `d`: Atom „d“ (directed)

Impl. der Paramodulation Einordnung in Tiefensuche mit BT

- gleichberechtigt zu anderen Regeln
  - keine Priorisierung
  - Reihenfolge der Regeln im Quelltext entscheidet
- `"="(a, b).`
  - zuerst `a` gesucht und durch `b` ersetzt
  - dann `b` gesucht und durch `a` ersetzt
- nur auf das erste Teilziel angewendet
- Argumente von links nach rechts durchsucht, Tiefensuche
- bei Funktionen: zuerst ganze Funktion, dann Argumente

Impl. der Paramodulation Probleme und Lösungen

- Problem:Reflexivität
  - Regel der ungeordneten Paramodulation wird immer wieder angewendet
  - Programm terminiert nicht
  - Regel mit Bedingungen: Bedingungen werden immer wieder der Hypothese hinzugefügt
- Lösung: Optimierungen in PHP-Prolog
  - Eliminierung von Teilzielen (Vermeidung der Expansion der Hypothese)
  - Speichern der Historie bereits erreichter Ziele (Vermeidung von unendlichen Schleifen)

Impl. der Paramodulation Probleme und Lösungen

- Problem: Paramodulation in Variablen
  - mögliche Expansion des Terms des Ziels
  - Programm terminiert nicht
- Lösung: verbieten
  - Paramodulation in Variablen nicht notwendig
  - Term mit allen anderen variablenfremden Termen unifizierbar

Impl. der Paramodulation Probleme und Lösungen

- Probleme:
  - Anwendbarkeit einer Regel auf ihre Bedingungen
  - Zusammenspiel mehrerer Regeln der Paramodulation
- kann zu nicht terminierenden Programmen führen
- Lösung:
  - Programmierer

Impl. der Paramodulation Einschränkungen

- immer
  - keine Paramodulation in Variablen
  - ersetzen eines Terms durch einen syntaktisch gleichen verboten

Impl. der Paramodulation Einschränkungen

- ungeordnete Paramodulation
  - `"="(a, b) :- p1, …, pn`
  - `a` kein Teilterm von `b` oder umgekehrt
  - $p_1, \ldots, p_n$: keine Aufrufe von Prädikaten mit Nebeneffekten
  - keine neuen Variablen

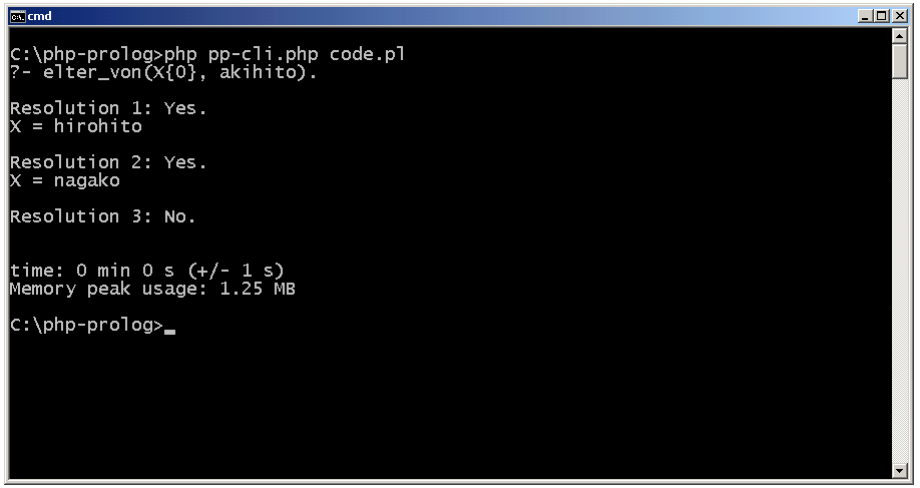
Impl. der Paramodulation Einschränkungen

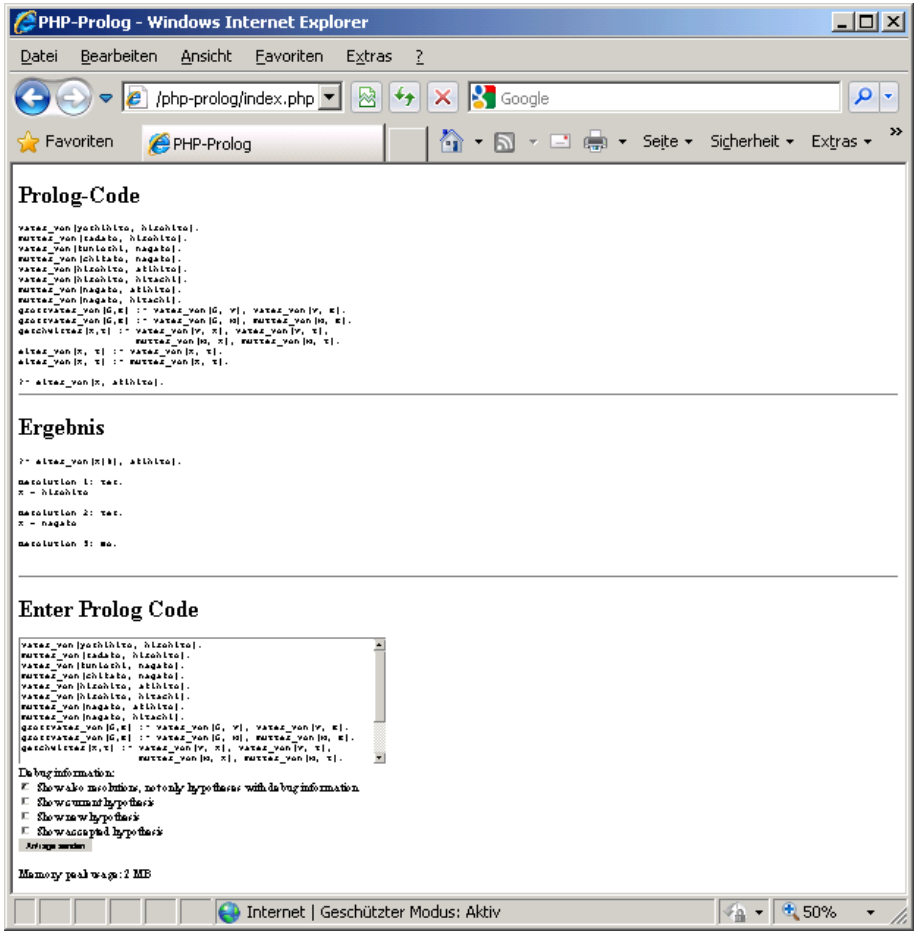
- geordnete Paramodulation
  - `"="(a, b) :- p1, …, pn`
  - `a` ist echter Teilterm von `b` oder umgekehrt
  - Terme in Köpfen von Programmklauseln: Normalform empfohlen

Impl. der Paramodulation Einschränkungen

- gerichtete Paramodulation
  - `"="(a, b, d) :-` p_1`, …,` p_n
  - `b` ist Teilterm von `a` sollte vermieden werden
  - Terme in Köpfen von Programmklauseln: Normalform empfohlen

Demo


cmd
C:\php-prolog>php pp-cli.php code.pl
?- elter_von(X{0}, akihito).
Resolution 1: Yes.
X = hirohito
Resolution 2: Yes.
X = nagako
Resolution 3: No.
time: 0 min 0 s (+/- 1 s)
Memory peak usage: 1.25 MB
C:\php-prolog>_


PHP-Prolog - Windows Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
/php-prolog/index.php
Google
Favoriten
PHP-Prolog
Seite Sicherheit Extras
Prolog-Code
Ergebnis
Enter Prolog Code
Debug information:
Show also resolutions, not only hypotheses with debug information
Show current hypothesis
Show new hypothesis
Show accepted hypothesis
Memory peak usage: 2 MB
Internet | Geschützter Modus: Aktiv
50%

Fazit und Ausblick

- Fazit
  - Paramodulation in Deduktionstool realisierbar, aber Einschränkungen
- Ausblick
  - Vorverarbeitung
  - Idempotenz und Vergleich der Hypothesen mit Variablenumbenennung
  - Schleifenerkennung für interaktive Programme
  - nur geordnete Paramodulation
  - ausführlicher Test von PHP-Prolog

Danke für die Aufmerksamkeit

Fragen?

Q&A