Heidelberg, den 6.2.2009

Answer Set Programming

mit Answer Set Prolog (A-Prolog)

Wangler Thomas
Logikprogrammierung
Institut für Computerlinguistik
Universität Heidelberg

Inhaltsverzeichnis

Inhaltsverzeichnis

1. Einführung

1.

1.1 Was ist Answer Set Programming?

- Eine Art der deklarativen (Logik)-Programmierung
- Beschäftigt sich vornehmlich mit Suchproblemen
- Basiert auf 'stable model semantics' (= answer set)
- Schafft Problemumgebungen, in denen 'answer set solver' suchen

1. Einführung

1.

1.2 Woher kommt Answer Set Programming?

- Automatisches Planen (Dimopoulos, Nebel und Köhler)
- Produktkonfiguratoren (Soininen und Niemelä)
- Bezeichnung stammt von Marek und Truszczyński / Niemelä (1999)
- Von Marek und Truszczyński zu einem Paradigma erhoben

1. Einführung

1.

1.3 Wofür benutzt man Answer Set Programming?

- Automatisches Planen
- Suche
- Folgern (reasoning)

2. Answer Set Prolog

2.1 Was ist Answer Set Prolog (A-Prolog)?

- Sprache auf Prolog-Basis für :
  - Wissensrepräsentation
  - Automatisches Schließen
- Basierend auf 'answer set semantics'
- Nutzt disjunkte Datenbanken und nicht-monotone Logik
- Modellierung mit 'Defaults' und 'Exceptions'

2. Answer Set Prolog

2.2 A-Prolog Syntax & Semantik

Ein A-Prolog Programm ist ein Paar $\{\sigma, \Pi\}$, wobei σ die 'signature' und Π eine Menge von Regeln über σ ist.

Eine A-Prolog Regel hat die Form:

l_0 or . . . or $l_k \leftarrow l_{k+1}, \ldots, l_m$, not $l_{m+1}, \ldots,$ not l_n

2. Answer Set Prolog

2.2 A-Prolog Syntax & Semantik

'*A partial interpretation S of σ(Π) is an* answer set *for Π if S is minimal (in the sense of set-theoretic inclusion) among the partial interpretations satisfying the rules of Π.'*

Ein Answer Set ist also eine minimale Menge von Regeln S, die die Gesamtregeln in Π wahrheitsgemäß erfüllen.

// Anm: Suche noch eine gute deutsche Definition...

2. Answer Set Prolog

2.3 A-Prolog Answer Set Beispiele

Π_1 = {q(a) or q(b).} hat als Answer Sets {q(a)} und {q(b)}.

Π_2 = { q(a) or q(b). ; ¬q(a). }

hat als Answer Set {¬q(a), q(b)}.

2. Answer Set Prolog

2.3 A-Prolog Answer Set Beispiele

$$\Pi_2 = \begin{cases} p(a) \leftarrow q(a). \\ p(a) \leftarrow \neg q(a). \end{cases}$$

hat als Answer Set ?

2. Answer Set Prolog

2.3 Die Unterschiede zu 'gängigen' Prolog-Implementierungen

- 2.3.1 (epistemische) Disjunktion
- 2.3.2 'klassische' Negation (negation as failure)
- 2.3.3 Regeln mit leeren Köpfen (constraints)

2. Answer Set Prolog

2.3.1 (epistemische) Disjunktion

In A-Prolog kann man im Gegensatz zum 'klassischen' Prolog die Disjunktion 'or' benutzen. Disjunktionen dieser Art sehen so aus:

q(a) or q(b) ← *q(c)*

2. Answer Set Prolog

2.3.2 'klassische' Negation (negation as failure)

Neben der Disjunktion kennt A-Prolog auch die klassische Negation 'not'.

q(a) ← q(b), not q(c)

2. Answer Set Prolog

2.3.3 Regeln mit leeren Köpfen (Constraints)

In A-Prolog ist eine Regel mit einem leeren Kopf ein Constraint (Restriktion).

head(r) = {}

$\leftarrow l_{k+1}, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$

3. Ein Beispiel

3. Ein Beispiel einer Wissensrepräsentation in A-Prolog

Wir nehmen an, es gäbe an einer Universität *h* ein kleines Institut *coli* im linguistischen Seminar *lingS*. Das Institut möchte für das nächste Semester den Kursplan erstellen.
Dafür erstellen wir eine Wissensrepräsentation *W.*

Annahme der Einfachheit halber: eine unendlich große 'signature'-Liste.

3. Ein Beispiel

3.1 'facts'

member(joe, *coli*). *member*(nils, *coli*). *member*(matthew, *coli*).

course(*java, coli*). *course*(syntax, *coli*). *course*(*ki, coli*).

course(*logic, coli*).

3. Ein Beispiel

3.2 'closed world assumptions'

¬member(*P, coli*) ← *not member*(*P, coli*).

¬course(*C, coli*) ← *not course*(*C, coli*).

3.3 Vorläufiger Kursplan

teaches(nils, *logic*).

teaches(matthew, *java*).

3. Ein Beispiel

3.4 Queries?

?- member(beate*, coli*).

?- teaches(beate, ki).

3. Ein Beispiel

3.5 Queries?

?- member(beate*, coli*). - no

?- teaches(beate, ki). - unknown

3. Ein Beispiel

3.6 Zusatzregeln

¬teaches(P, C) ← *¬member(P, coli),*
course(C, coli),
not ab(d1(P, C)),
not teaches(P, C).

ab(d1(P, ki)) ← *not ¬member(P, info).*

Was bedeuten diese Regeln?

3. Ein Beispiel

3.7 Queries zum Zweiten?

member(beate, info).

3.

?- teaches(beate*, syntax*). - no

?- teaches(beate, ki). - unknown

3. Ein Beispiel

3.8 Weitere Regeln

¬teaches(P1, C) ← *teaches(P2, C),*
P1 =! P2,
not ab(d2(P1, C)),
not teaches(P1, C).

Die Wissensrepräsentation erlaubt jetzt auch starke Ausnahmen (Ausnahmen, die die Folgerung einer Regel außer Kraft setzen)

teaches(john, ai).

3. Ein Beispiel

3.9 Weitere Regeln

part(linguS, *coli*).
part(*coli*, *h*).
part(*E*1, *E*2) ← *part*(*E*1, *E*), *part*(*E*, *E*2).
¬*part*(*E*1, *E*2) ← *not part*(*E*1, *E*2).
member(*P*, *E*1) ← *part*(*E*2, *E*1), *member*(*P*, *E*2).

Zusammen mit der 'closed world assumption'
¬*member*(*P*, *Y*) ← *not member*(*P*, *Y*).
kann die Wissenrepräsentation jetzt z.B. die Anfrage *?- member(nils,h).* beantworten.

3. Ein Beispiel

3.10 Eine Relation

offered(C, D) ← *course(C, D), teaches(P, C).*

¬offered(C, D) ← *course(C, D), not offered(C, D).*

teaches(tom, ki) or teaches(matthew, ki).

Damit ist unser kleines Beispiel zu Ende und ein answer set solver könnte auf dieser Wissensrepräsentation suchen.

4. Anwendungen

4. Anwendungen

- Antriebssteuerung in Space Shuttles (USA-Advisor)
- Künstliche Intelligenz
- Datenbanken
- Wirtschaft (product configuration)
- Geologie, Zoologie, Linguistik

5. Zusammenfassung

5. Zusammenfassung

- A-Prolog ist geeignet für Answer-Set-Umgebungen
- Klare Prolog Syntax und Semantik
- Sehr schnell in der automatischen Suche
- einfach beweisbare, effiziente und schnelle Programmierung

5. Quellen

- http://en.wikipedia.org/wiki/Answer_set_programming (3.2.09)
- http://en.wikipedia.org/wiki/Stable_model_semantics (3.2.09)

- Michael Gelfond. Answer sets. In Handbook of Knowledge Representation. Elsevier Science, 2007.

- Marcello Balduccini, Michael Gelfond, and Monica Nogueira. Answer set based design of knowledge systems. Annals of Mathematics and Artificial Intelligence, 2006.

Vielen

Dank

für

die

Aufmerksamkeit