Logikorientierte Programmierung
Prüfungsfragen vom 13. 10. 2011

3. Februar 2012

Wenn ihr einen Fehler gefunden habt, meldet euch bitte:
`e0906307@student.tuwien.ac.at`

1. (a) Erklären Sie die Wissensrepräsentation in Prolog. Wie kann positives bzw. negatives Wissen dargestellt werden. Wie sieht eine Anfrage in Prolog auf dieses Wissen aus?

   (b) Terminiert die Rückwärtssuche in $\texttt{GOL}^+$ immer? Falls ja, geben Sie eine ausführliche Erklärung an, falls nein, bitte eine genaue Beschreibung, wie die Suche terminierend gemacht werden kann.

   (c) Untersuchen Sie, ob das Absorptionsgesetz

   $$a \leftrightarrow a \wedge (a \vee b)$$

   in $\texttt{GOL}^+$ beweisbar ist. Hinweis: Es ist u.U. ein Transformationsschritt nötig, um die Formel in das richtige Format zu bringen.

   Zur Erinnerung: Neben Axiomen umfasst $\texttt{GOL}^+$ folgende Regeln:

   $$\frac{\vdash e, N}{\vdash e \vee f, N} \vee$$

   $$\frac{\vdash f, N}{\vdash e \vee f, N} \vee$$

   $$\frac{\vdash e, N \qquad \vdash f, N}{\vdash e \wedge f, N} \wedge$$

   $$\frac{\vdash N}{\vdash e, N} W$$

2. (a) Beschreiben Sie die allgemeine Idee der Answer-Set-Programmiermethodologie.

   (b) Sei I eine Menge von Grundliteralen und P ein grundiertes Programm.

      i. Definieren Sie den Begriff des Reduktes P^I.

      ii. Wann ist I ein Answer Set von P?

   (c) Gibt es ein disjunktives logisches Programm P, sodass P Answer Sets X_1, X_2 besitzt, welche die Bedingung $X_1 \subset X_2$ erfüllen, d.h. sodass X_1 eine *echte* Teilmenge von X_2 ist?

   (d) Betrachten Sie folgendes Programm (a, b, c sind Grundatome):

$$P = \{\neg c \vee \neg d \texttt{ :- } \\ a \vee b \texttt{ :- not } c.\}$$

   Geben Sie eine Menge Q von Constraints an, sodass $P \cup Q$ zwei Answer Sets, $\{\neg c, a\}$ und $\{\neg c, b\}$, besitzt.

3. (a) Betrachten Sie Eingabefakten, wie sie zum Testen des 3. Übungsbeispieles verwendet wurden. Insbesondere seien Ihnen Fakten der Form `paper(P)` (P ist ein Paper) und `assigned(P, M)` (P wurde PC-Member M zugeteilt) in Erinnerung gerufen.

   Weiters sei für jeden PC-Member auch ein Fakt der folgenden Form inkludiert:

      `bid_at(M, T)`: PC-Member M hat seine Gebote zum Zeitpunkt T abgegeben;

   wobei T eine Ganzzahl ist, die einen *eindeutigen* Zeitpunkt repräsentiert, d.h., zeitgleiche Gebote sind nicht möglich.

   Repräsentieren Sie folgende Sachverhalte durch logische Programmregeln in `DLV`-Syntax unter Verwendung von Aggregatatomen und Weak Constraints:

      i. Definieren Sie ein Prädikat `order(M, O)`, welches jedem PC-Member M jene Ordinalzahl (beginnend mit 1) zuweist, die das Auftreten des Gebotes von M in der zeitlichen Reihenfolge der Gebote repräsentiert. Beispielsweise soll `order(m, 3)` repräsentieren, dass `m` als dritter PC-Member sein Gebot abgegeben hat.

      ii. Definieren Sie ein Prädikat `timing(P, S)`, welches jedem Paper P die Summe S der Ordinalzahlen der dem Paper zugewiesenen PC-Member zuordnet.

iii. Formalisieren Sie einen Penalty für die zu evaluierende Eingabe, indem Sie für jedes Paper P mit `timing` größer oder gleich 40 einen Penalty auf Level 3 vergeben; der Penalty soll die Differenz zwischen Ps `timing` und 4 sein.

(b) Markieren Sie richtige Antworten. Sei M eine Menge von Grundliteralen und P ein grundiertes Programm. Für das *generalisierte Redukt* (bez. Aggregatatome) gilt, dass

- eine Regel nicht gelöscht wird, wenn im positiven Rumpf ein Aggregatatom a vorkommt, welches bez. M wahr ist.
- eine Regel nicht gelöscht wird, wenn im negativen Rumpf ein Aggregatatom a vorkommt, welches bez. M falsch ist.
- wenn eine Regel nicht gelöscht wird, dann werden alle Aggregatatome daraus entfernt.
- wenn eine Regel nicht gelöscht wird, dann werden alle Aggregatatome daraus entfernt, die bez. M wahr sind.

(c) Gegeben seien folgende Programme und Weak Constraints (a, b, c, d sind Grundatome):

$$P = \{a \vee c. \\ d \text{ :- } a, c.\}$$

$$Q = \{b \text{ :- } \mathit{not}\ c, \mathit{not}\ d. \\ c \text{ :- } \mathit{not}\ b, \mathit{not}\ \neg d.\}$$

$$R = \{b \text{ :- } a, \mathit{not}\ d. \\ b \text{ :- } c.\}$$

$$W_1 = \{\ :\sim c.\ [4:1] \\ :\sim b.\ [1:1] \\ :\sim b.\ [3:1]\}$$

$$W_2 = \{\ :\sim b. \\ :\sim b, c. \\ :\sim a, b, c.\}$$

Geben Sie die Best Models mit den entsprechenden Kosten (weight und level) der folgenden Programme mit Weak Constraints an:

i. $P \cup Q, W_1$

ii. $P \cup R, W_2$