Humboldt-Universität zu Berlin
Institut für Informatik
Lehrstuhl Logik in der Informatik
Prof. Dr. Nicole Schweikardt

12. November 2014

Logik in der Informatik

Wintersemester 2014/2015

Übungsblatt 4

Abgabe: *bis 19. November 2014, 9.15 Uhr (vor der Vorlesung oder im Briefkasten zwischen den Räumen 3.401 und 3.402 im Johann von Neumann-Haus (Rudower Chaussee 25))*

Aufgabe 1: (30 Punkte)

Seien P, Q, R, S, T, U unterschiedliche Aussagensymbole aus AS.

(a) Geben Sie für folgende Klauselmengen jeweils eine erfüllende Interpretation oder eine Resolutionswiderlegung an:

(i) $\Gamma_1 := \{\{P, \neg S\}, \{S\}, \{S, R\}, \{\neg S, \neg P\}\}$

(ii) $\Gamma_2 := \{\{P, Q, \neg R\}, \{\neg T\}, \{P, Q, \neg S\}, \{\neg Q, P, \neg R\}, \{\neg S, \neg Q, \neg P, R\}, \{S, \neg S\}, \{Q, \neg S, R\}\}$

(iii) $\Gamma_3 := \{\{\neg P, Q, R, S\}, \{P, R, \neg S\}, \{\neg P, \neg Q\}, \{\neg R, S\}, \{P\}, \{\neg P, \neg S\}\}$

(b) Wenden Sie den DPLL-Algorithmus auf die folgende Klauselmenge Γ an. Erklären Sie dabei Schritt für Schritt, wie der Algorithmus vorgeht.

$$\Gamma := \Big\{\{\neg P, R, S\}, \{\neg Q, U, S\}, \{\neg Q, \neg U, \neg R\}, \{\neg S, \neg Q\}, \{Q, \neg R, \neg P\}, \{Q, U, R\}, \{Q, \neg U, \neg S\}, \{P, T\}, \{P, U\}, \{\neg U, R, \neg T\}, \{P, \neg R, \neg T\}\Big\}$$

Aufgabe 2: (20 Punkte)

Beim Assessment-Center zu Ihrer Bewerbung auf Ihren absoluten Traumjob bekommen Sie die Aufgabe gestellt, innerhalb eines Nachmittags Programme zu erstellen, die die Probleme

- *Allgemeingültigkeit:* Die Eingabe besteht aus einer beliebigen aussagenlogischen Formel in ASCII-Repräsentation (gemäß Folie 49 des Skripts zur Vorlesung), und die Frage ist, ob die eingegebene Formel allgemeingültig ist.
- *Folgerung:* Die Eingabe besteht aus zwei aussagenlogischen Formeln in ASCII-Repräsentation, und die Frage ist, ob die zweite Formel aus der ersten folgt.
- *Äquivalenz:* Die Eingabe besteht aus zwei ausssagenlogischen Formeln in ASCII-Repräsentation, und die Frage ist, ob die beiden Formeln äquivalent sind.

mit atemberaubender Geschwindigkeit lösen. Am Ende des Tages werden Ihre Programme gegen die Programme Ihrer Mitbewerber antreten, und zwar für eine Reihe Ihnen unbekannter aussagenlogischer Formeln, die mehrere Tausend aussagenlogische Symbole enthalten können. Naheliegenderweise wird derjenige Bewerber den Job bekommen, dessen Programme am schnellsten laufen und stets die korrekten Antworten geben.

Ihnen ist natürlich sofort klar, dass es keine gute Idee ist, Ihre Programme "from scratch" zu schreiben. Glücklicherweise dürfen Sie auf die im Internet frei verfügbaren Programme zugreifen, die an der *SAT Competition 2014*[1] teilgenommen haben. Blöderweise lösen diese Programme allerdings nicht die für Sie relevanten Probleme, sondern „nur" das aussagenlogische Erfüllbarkeitsproblem für Formeln in konjunktiver Normalform, die im speziellen DIMACS-Eingabeformat gegeben sind.

[1] `http://satcompetition.org/2014/`

(a) Beschreiben Sie, wie Formeln in konjunktiver Normalform im DIMACS-Format repräsentiert werden (Informationen dazu finden Sie im Internet, u.a. bei der Beschreibung der Regeln der SAT Competition 2014).

(b) Nehmen Sie an, dass Sie ein sehr effizientes Programm SAT-SOLVER zur Verfügung haben, das bei Eingabe einer Text-Datei `formel.cnf`, die die DIMACS-Repräsentation einer KNF-Formel enthält, als Ausgabe das Wort SATISFIABLE bzw. UNSATISFIABLE liefert, je nach dem, ob die eingegebene KNF-Formel erfüllbar ist oder nicht. Entwerfen Sie Algorithmen, die unter Verwendung dieses Programms die im Assessment-Center gestellten Probleme lösen (und zwar einerseits möglichst effizient und andererseits so, dass Sie den Programm-Code innerhalb eines Nachmittags erstellen können).

(c) Geben Sie eine Abschätzung der Laufzeit Ihrer Algorithmen an. Gehen Sie dabei davon aus, dass $f_{\text{SAT-SOLVER}} : \mathbb{N} \to \mathbb{N}$ eine Funktion ist, so dass das Programm SAT-SOLVER bei Eingabe einer Datei der Länge n höchstens $f_{\text{SAT-SOLVER}}(n)$ viele Berechnungsschritte durchführt, und verwenden Sie diese Funktion, um eine Abschätzung der Laufzeit Ihrer Programme bei Eingabe von Formeln der Längen n_1 und n_2 anzugeben.

Aufgabe 3: (25 Punkte)

(a) Wenden Sie den Streichungsalgorithmus auf die Formel φ an:

$$\varphi := \quad P \wedge R \wedge (\mathbf{1} \to T) \wedge (\neg P \vee \neg Y) \wedge \Big((P \wedge R) \to S\Big) \\ \wedge \Big((S \wedge X) \to Y\Big) \wedge (\neg P \vee \neg T \vee X) \wedge (S \vee \neg Y)$$

Formen Sie die Formel zunächst um, so dass sie eine passende Eingabe für den Algorithmus ist. Erklären Sie Schritt für Schritt, wie der Algorithmus bei Eingabe der Formel vorgeht.

(b) **(i)** Geben Sie eine Formel $\varphi \in \mathsf{AL}$ an, die zu keiner Hornformel äquivalent ist.

(ii) Gibt es eine Formel in AL, die zu keiner Hornformel erfüllbarkeitsäquivalent ist?

Beweisen Sie jeweils, dass Ihre Antwort korrekt ist.

Aufgabe 4: (25 Punkte)

Lesen Sie Kapitel 4 aus dem Buch "Learn Prolog Now!".

(a) Wie antwortet Prolog auf die folgenden Anfragen?

(i) `?- [a, b] = [X, Y].`

(ii) `?- [X | []] = [c].`

(iii) `?- [[] | [b, c]] = [X, _, Z].`

(iv) `?- [H | T] = [a, b | [c | [d]]].`

(b) Das Prädikat `member/2` wird in Abschnitt 4.2 des Buchs "Learn Prolog Now!" definiert. Zeichnen Sie den Suchbaum für die Anfrage `?- member(2, [1, X]).`!

(c) Definieren Sie *rekursiv* ein Prädikat `nimm/3`, so dass `nimm(E, X, Y)` genau dann erfolgreich ist, wenn `E` ein Element der Liste `X` ist und `Y` aus der Liste `X` durch Löschung eines Vorkommens von `E` entsteht.

(d) Wir kodieren aussagenlogische Literale wie folgt durch Prolog-Terme: Ist `X` $\in \mathbb{N}$ mit $X \geq 1$, dann repräsentiert `pos(X)` das Literal $V_{\texttt{X}}$ und `neg(X)` das Literal $\neg V_{\texttt{X}}$. Weiterhin kodieren wir Mengen von Literalen als Prolog-Listen. Beispielsweise repräsentieren wir $\{V_1, \neg V_2, \neg V_3\}$ durch `[pos(1), neg(2), neg(3)]`.

Schreiben Sie ein Prädikat `resolvente/3`, so dass Folgendes gilt: Unter der Annahme, dass `L1`, `L2` und `R` Mengen von Literalen repräsentieren, ist `resolvente(L1, L2, R)` erfüllt wenn `R` eine Resolvente von `L1` und `L2` ist. Beispielsweise sollte die Anfrage

```
?- resolvente([pos(1), neg(3), pos(4)], [pos(2), pos(3), neg(4)], R).
```

zu folgenden Ausgaben führen:

`R = [pos(1),pos(4),pos(2),neg(4)]` und `R = [pos(1),neg(3),pos(2),pos(3)]`

Hinweise: Nutzen Sie gegebenenfalls das Prädikat `nimm/3` aus Teilaufgabe **(c)**. Haben Sie **(c)** nicht gelöst, so können Sie die Online-Hilfe von SWI-Prolog nutzen, um sich mit dem vordefinierten Prädikat `select/3` vertraut zu machen. Nutzen Sie außerdem das vordefinierte Prädikat `union/3`.