Dr. Christian Săcărea

"Babeş–Bolyai" Universität, Cluj-Napoca
Fakultät für Mathematik und Informatik

Wintersemester 2020/2021

6. Übung zur Vorlesung

Logik für Informatiker

Gruppenübungen:

(G 1)Hornformeln

Sei $\mathcal{P} := \{P, Q, R, S, U\}$ eine Menge von Aussagenvariablen. Sei F die folgende aussagenlogische Formel über $\mathcal{P}$:

$$F := (\neg P \vee \neg Q \vee \neg U \vee R) \wedge Q \wedge (\neg Q \vee P) \wedge (\neg Q \vee \neg P \vee U) \wedge (\neg S \vee R).$$

a) Ist F eine Horn-Formel?

b) Schreiben Sie die gegebene Formel als Konjunktion von Implikationen auf.

c) Wenden Sie den Markierungsalgorithmus zur Überprüfung der Erfüllbarkeit von Hornklauseln auf die im Schritt b) erzeugte Konjunktion von Implikationen an. Geben Sie explizit für jeden Schritt an, welche Atome markiert werden, und wieso die Markierung zustande kommt.

d) Ist die Formel F erfüllbar oder unerfüllbar? Begründen Sie kurz Ihre Antwort mit Hilfe der Ergebnisse aus c); falls F erfüllbar ist, geben Sie ein Modell an.

(G 2)Hornformeln - Textaufgabe

Ich habe Kartoffeln, Öl und Reis zu Hause, aber Lust auf Sushi.Und jetzt? Meine japanische Nachbarin wird mir, wenn ich ihr sowohl Rindfleisch als auch Pommes Frites vorbeibringe, etwas Wasabi geben. Und aus Kartoffeln und Öl kann ich Pommes Frites und Kartoffelsalat machen, soviel von beidem wie ich will. Und für Sushi brauch' ich eigentlich nur Fisch und Reis und Wasabi. Aber mein Nachbar, der seit neustem eine vegetarische Freundin hat, gibt mir sicher Fisch und Rindfleisch, wenn ich genug Reis und Kartoffelsalat vorbeibringe. . .

Kann ich jetzt also Sushi machen oder nicht?

(G 3)DPLL

Wende den DPLL Algorithmus um die Erfüllbarkeit der folgenden Formel zu testen:

$$(R) \wedge (\neg P) \wedge (P \vee R) \wedge (Q \vee \neg R \vee S) \wedge (\neg Q \vee \neg R \vee S) \wedge (P \vee \neg Q \vee \neg R \vee \neg S)$$

(G 4)SAT-Solver

Mache dich mit folgenden SAT-Solvern vertraut:

- Minisat: http://minisat.se/
- SAT4J ist in Java implementierter SAT-Solver; leicht als Library statt über DIMACS-Eingabe zu bedienen http://www.sat4j.org/
- Picosat: http://fmv.jku.at/picosat/
- Berkmin: http://eigold.tripod.com/BerkMin.html
- RSat: http://reasoning.cs.ucla.edu/rsat/

- zChaff: http://www.princeton.edu/~chaff/zchaff.html

SAT-Solver verlangen typischerweise eine Eingabe in KNF. Es gibt ein standardisiertes Format: DIMACS

- Variablen sind natürliche Zahlen ≥ 1
- Literale werden durch Integer bezeichnet, z.B. $A_7 = 7, \neg A_4 = -4$
- Klausel ist Liste von Integern, 0 markiert Klauselende
- KNF ist Liste von Klauseln
- Kommentare im Header haben das Format `(c ...)`
- spezielle Headerzeile `(p cnf ...)` gibt Anzahl verwendeter Klauseln und Variablen an

Beispiel: Die KNF

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

kann im DIMACS-Format so repräsentiert werden:

```
-1 2 3 0
2 -3 0
-4 0
1 4 0
-2 -3 -4 0
```

Clevere Heuristiken und jahrelanges Tuning haben dazu geführt, dass moderne SAT-Solver typischerweise Instanzen der Größenordnung 10^5 Variablen und 10^6 Klauseln lösen können.

Vorsicht! Es gibt natürlich auch (im Vergleich dazu) sehr kleine Instanzen, an denen sie sich die Zähne ausbeissen.

Typischer Einsatz von SAT-Solvern:

- Hardware-Verifikation
- Planungsprobleme in der KI
- Constraint-Solving
- ...