Uwe Waldmann Saarbrücken, 22.6.1999

Übungen zur Vorlesung

Praxis des Programmierens

Projekt

1 Aufgabenstellung

Ziel des Projektes ist die Entwicklung und Implementierung zweier Programme zur Lösung des aussagenlogischen Erfüllbarkeitsproblems auf der Basis des Davis-Putnam-Algorithmus. Das erste der beiden Programme hat die Aufrufsyntax

`java DavisPutnam` [`-t` *k*] *Dateiname*

Es akzeptiert Eingabedateien im DIMACS-cnf-Format (siehe unten). Die Option „`-t` *k*" (*k* eine natürliche Zahl in Dezimaldarstellung) beschränkt die Laufzeit des Programms auf *k* Sekunden. Die Ausgabe hat folgende Form:

`Problem:` *Dateiname*
`Result:` *ResString*
[`Solution:` [`-`] *Variable* [`-`] *Variable* ...]

Im Falle eines erfüllbaren Problems ist *ResString* der String „`yes`", bei einem unerfüllbaren Problem „`no`" und bei einem Abbruch durch die Zeitschranke „`ran out of time`". Eine Belegung wird nur bei einer erfüllbaren Klauselmenge ausgegeben; in der Belegung steht „$-i$" für „$A_i = 0$" und „i" für „$A_i = 1$".

Das zweite der beiden Programme hat die Aufrufsyntax

`java VisualDavisPutnam` *Dateiname*

Es unterscheidet sich von `DavisPutnam` dadurch, daß der Benutzer den Algorithmus im Einzelschrittmodus ablaufen lassen kann, wobei die Zwischenzustände der Klauselmenge während des Algorithmus in einem neu geöffneten Fenster angezeigt werden. Die Anzeige sollte in einer Form erfolgen, die es dem Benutzer erlaubt, einen Einblick in die Funktionsweise des Algorithmus zu bekommen. (Zusätzliche Anzeigen und Interaktionsmöglichkeiten für den Benutzer sind zulässig, aber nicht erforderlich.)

2 Dateiformat

Eine Datei im DIMACS-cnf-Format enthält (in dieser Reihenfolge)

- optional: beliebig viele Kommentarzeilen, die jeweils mit dem Kleinbuchstaben „`c`" beginnen.
- eine Problemzeile. Diese hat die Form „`p cnf` *n* *m*", wobei *n* die Anzahl der Variablen und *m* die Anzahl der Klauseln ist.
- *m* Klauseln über den Variablen A_1 bis A_n.

Eine Klausel wird kodiert als Folge von ganzen Zahlen, wobei i für A_i und $-i$ für $\neg A_i$ steht. Eine Klausel wird durch den Wert 0 beendet; bei der letzten Klausel kann die abschließende 0 entfallen. Die einzelnen Zahlen werden getrennt durch Leerzeichen, Tabs und/oder Newlines;

Klauseln dürfen sich über mehrere Zeilen erstrecken. Es braucht nicht jede in der Problemzeile deklarierte Variable wirklich in einer der Klauseln vorkommen.

Die Klauselmenge $(A_1 \vee A_3 \vee \neg A_5) \wedge (A_5) \wedge (\neg A_3 \vee A_2)$ könnte also zum Beispiel durch die folgende Eingabedatei repräsentiert werden:

```
c Example CNF format file
c
p cnf 5 3
1 3 -5 0
5 0 -3
2
```

3 Vorgehensweise

1. Das Verhalten von `VisualDavisPutnam` ist nur in groben Umrissen vorgegeben. Spezifiziert das Verhalten dieses Programms. Was soll dargestellt werden? Wie soll es dargestellt werden? Welche Interaktionsmöglichkeiten hat der Benutzer?

2. Entwerft geeignete Klassen zur Implementierung von `DavisPutnam` und `VisualDavisPutnam`. Dabei sollen beide Programme mit einer gemeinsamen Klassenmenge implementiert werden. Ein hohes Maß an Code-Wiederverwendung ist dabei ebenso erwünscht wie eine möglichst effiziente Implementierung von `DavisPutnam`.

3. Stellt Euren Entwurf in einer der Übungen in der Woche vom 12. bis zum 16. Juli in einem maximal fünfminütigen Vortrag vor.

4. Implementiert Euren Entwurf, dokumentiert ihn (mit LaTeX) und erstellt eine kurze Benutzeranleitung für `VisualDavisPutnam` (ebenfalls mit LaTeX).

5. Schickt Eure Implementierung, die Dokumentation Eures Entwurfs und die Benutzeranleitung bis zum 16. August, 24:00 Uhr an `uwe@mpi-sb.mpg.de`. (Details zum Format der Mail folgen.) Seid für Rückfragen, Nachbesserungen, etc. in den folgenden zwei Wochen erreichbar.

4 Qualitätskriterien

Die Aufgabe gilt als gelöst, sofern folgende Bedingungen erfüllt sind:

1. Die geforderte Funktionalität ist eingehalten.

2. Die Programme sind korrekt. Insbesondere verhalten sie sich korrekt auf allen Testbeispielen. `VisualDavisPutnam` verhält sich so, wie in der Benutzeranleitung beschrieben.

3. Die Programme sind lesbar. Die in der Vorlesung besprochenen Namens- und Formatierungskonventionen sind eingehalten. Jede Klasse, jede Methode und jedes Interface ist dokumentiert (ausgenommen solche, die trivial *und* privat sind). Die Autoren der einzelnen Komponenten sind angegeben.

4. Die Dokumentation des Entwurfs (in LaTeX, ca. 5 Seiten) enthält eine begründete Beschreibung der getroffenen Designentscheidungen. (Zeichnungen sind nicht erforderlich.)

5. Keine groben Fouls.