Ludwig-Maximilians-Universität München Institut für Informatik

Dr. M. Hölzl, C. Kroiß

Formale Techniken der Software-Entwicklung Übungsblatt 7 Besprechung am 19.06.2015

Musterlösung

Aufgabe 1:

In der Vorlesung wurde eine einfache Theorie für den Snark-Theorembeweiser vorgestellt, mit der sich Listen ähnlich wie in z.B. ACL2 beschreiben lassen. Insbesondere findet man in der Datei snark-prog.lisp im Folien-Repository die folgenden Aussagen:

Hierbei sind car, cdr und member jeweils 2-stellige Prädikate und cons ein Funktionssymbol. Diese Aussagen können in reiner Logik-Schreibweise als Horn-Klauseln aufgefasst werden, d.h. als Klauseln, die jeweils nur ein positives Literal enthalten. Das ergibt die Klauselmenge \mathcal{K}^1 :

```
 \begin{split} \mathcal{K} = & \{ \operatorname{car}(\operatorname{cons}(x,xs),x), \\ & \operatorname{cdr}(\operatorname{cons}(x,xs),xs), \\ & \operatorname{member}(x,\operatorname{cons}(x,xs)), \\ & \operatorname{member}(x,xs) \implies \operatorname{member}(x,\operatorname{cons}(y,xs)) \, \} \end{split}
```

(a) Geben Sie ausgehend von den Klauseln in \mathcal{K} zusätzliche Klauseln an, die die Semantik für folgenden Prädikate definieren:

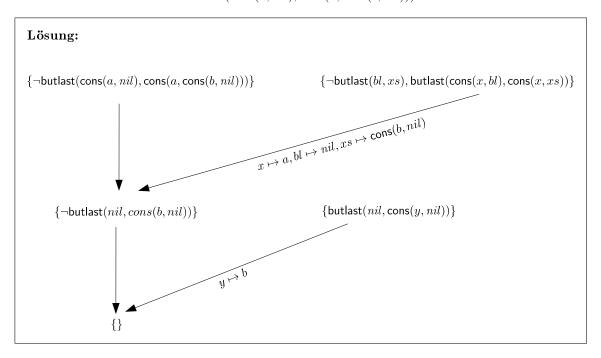
```
\begin{aligned} &\mathsf{last}(x,l) \mathbf{:} \\ &\mathsf{ist} \ \mathsf{wahr}, \ \mathsf{wenn} \ x \ \mathsf{das} \ \mathsf{letzte} \ \mathsf{Element} \ \mathsf{in} \ \mathsf{der} \ \mathsf{Liste} \ l \ \mathsf{ist}. \end{aligned} \mathsf{butlast}(p,l) \mathbf{:} \\ &\mathsf{ist} \ \mathsf{wahr}, \ \mathsf{wenn} \ p \ \mathsf{das} \ \mathsf{Pr\"{a}fix} \ \mathsf{der} \ \mathsf{Liste} \ l \ \mathsf{ohne} \ \mathsf{das} \ \mathsf{letzte} \ \mathsf{Element} \ \mathsf{ist}. \end{aligned} \mathsf{palindrome}(l) \mathbf{:} \\ &\mathsf{ist} \ \mathsf{wahr}, \ \mathsf{wenn} \ \mathsf{die} \ \mathsf{Liste} \ l \ \mathsf{ein} \ \mathit{Palindrom} \ \mathsf{darstellt}, \ \mathsf{also} \ \mathsf{z.B.} \ [A,B,C,B,A] \ \mathsf{oder} \ [A,B,B,A]. \end{aligned}
```

 $^{^1}$ Eigentlich is das vierte Element von $\mathcal K$ keine Klausel, weil eine Implikation enthalten ist. Die Darstellung hier ist jedoch etwas lesbarer als die tatsächliche Klauselform.

Lösung:

- $\bullet \ \, \mathsf{last}(x,\mathsf{cons}(x,nil))$
- $last(x, ys) \implies last(x, cons(y, ys)) \equiv {\neg last(x, ys), last(x, cons(y, ys))}$
- butlast(nil, cons(x, nil))
- $\mathsf{butlast}(bl,xs) \Longrightarrow \mathsf{butlast}(\mathsf{cons}(x,bl),\mathsf{cons}(x,xs))$ $\equiv \{\neg \mathsf{butlast}(bl,xs), \mathsf{butlast}(\mathsf{cons}(x,bl),\mathsf{cons}(x,xs))\}$
- palindrome(nil)
- palindrome(cons(x, nil))
- $\begin{array}{l} \bullet \quad (\mathsf{last}(x,xs) \, \wedge \, \mathsf{butlast}(ys,xs) \, \wedge \, \mathsf{palindrome}(ys)) \implies \mathsf{palindrome}(\mathsf{cons}(x,xs)) \\ \equiv \{\neg \mathsf{last}(x,xs), \neg \mathsf{butlast}(ys,xs), \neg \mathsf{palindrome}(ys), \mathsf{palindrome}(\mathsf{cons}(x,xs))\} \end{array}$
- (b) Verwenden Sie den prädikatenlogischen Resolutionskalkül, um folgende Aussage zu beweisen:

$$butlast(cons(a, nil), cons(a, cons(b, nil)))$$



Aufgabe 2:

Modellieren Sie ein einfaches Telefonbuch-System in ACL2. Dabei sollen folgende Funktionen unterstützt werden:

add-entry:

fügt einen Eintrag für einen angegebenen Namen mit einer angegebenen Telefonnummer zu einem Telefonbuch hinzu.

entry-existsp:

überprüft, ob ein Telefonbuch einen Eintrag für einen bestimmten Namen enthält.

get-number:

gibt die Telefonnummer für einen angegebenen Namen zurück.

(a) Spezifizieren Sie die oben angegebenen Features als ACL2-Funktionen.

- (b) Geben Sie ACL2-Theoreme an, um folgende Aussagen zu beweisen:
 - 1. Nach dem Hinzufügen eines Eintrags für einen bestimmen Namen mit add-entry ergibt ein Test mit entry-existsp, dass tatsächlich ein Eintrag mit diesem Namen im Telefonbuch existiert.
 - 2. Nach dem Hinzufügen eines Eintrags für einen Namen name mit der Nummer num ergibt die Suche mit get-number für den Namen name dieselbe Telefonnummer num.