Technische Universität München **WS 2007/08**
Esparza/Schwoon

Vorlesung Logik

Hinweise zu otter
(Erfüllbarkeitsprüfer für prädikatenlogische Formeln)

`otter` ist ein Tool, welches (neben anderen Dingen) das Resolutionsverfahren für prädikatenlogische Formeln implementiert. Mit anderen Worten kann man `otter` dazu verwenden, die Unerfüllbarkeit einer prädikatenlogischen Formel festzustellen. Wie wir wissen, ist dies im Alllgemeinen ein unentscheidbares Problem. Es kann also vorkommen, dass `otter` nicht terminiert, wenn eine Formel erfüllbar ist. Ist eine Formel unerfüllbar, so findet `otter` je nach Einstellung auf jeden Fall einen Beweis (braucht aber ggfs. sehr, sehr lange dafür) oder bricht die Suche vorzeitig ab, obwohl eine Formel unerfüllbar ist. Das Programm wendet jedoch etliche Heuristiken an, die dazu führen, dass in vielen Fällen eben doch ein Resolutionsbeweis gefunden wird, wenn einer existiert.
`otter` kann hier heruntergeladen werden:

`http://www.cs.unm.edu/∼mccune/otter/`

Im Lieferumfang befinden sich auch viele Beispiele sowie die Dokumentation, die alle Einstellungen erläutert. Im Folgenden geben wir nur eine kurze Einleitung, die zum Lösen einiger simpler Beispiele befähigt.
Syntax: `otter` akzeptiert nur Formeln in Skolem-Normalform, d.h. nur allquantifizierte Formeln, deren Matrix in KNF steht. Dabei werden die Allquantoren implizit angenommen, d.h. man schreibt einfach die Matrix der Formel hin.
Etwas verwirrend ist die Groß- und Kleinschreibung, da sie von der in der Vorlesung verwendeten Notation abweicht: Prädikate, Funktionen und Konstanten werden mit Klein-, Variablen mit Großbuchstaben bezeichnet.
Die Operatoren sind wie folgt: `-` steht für Negation und `|` für Disjunktion. Klauseln werden durch einen Punkt (`.`) getrennt.
Beispiel: Wir betrachten folgenden Syllogismus:

Einige M sind nicht P, alle M sind S, dann folgt: einige S sind nicht P.

Um die Gültigkeit dieses Syllogismus zu zeigen, übersetzen wir seine *Negation* (die dann ja unerfüllbar sein müsste) in eine prädikatenlogische Formel wie in Aufgabe 7.1 (b) (i). Diese wandeln wir in eine erfüllbarkeitsäquivalente Skolem-Normalform um und erhalten:

$$\forall y \forall z \big(M(a) \wedge \neg P(a) \wedge (\neg M(y) \vee S(y)) \wedge (\neg S(z) \vee P(z)) \big)$$

Diese Formel wird in `otter` wie folgt codiert:

```
set(prolog_style_variables).
set(binary_res).
clear(unit_deletion).
clear(factor).

list(sos).

m(a).
-p(a).
-m(Y) | s(Y).
-s(Z) | p(Z).

end_of_list.
```

Die ersten fünf Zeilen sind Meta-Anweisungen an `otter`, die im Wesentlichen besagen, dass es auf die nachfolgenden Formel das prädikatenlogische Resolutionsverfahren anwenden soll, wie wir es aus der Vorlesung kennen. Die eigentliche Formel folgt in den vier Zeilen danach.

Anwendung:
Der Aufruf erfolgt durch:

`otter <` *Dateiname*

Falls es `otter` gelingt, die Unerfüllbarkeit zu zeigen, gibt es einen Resolutionsbeweis aus (neben einigen anderen Diagnose-Ausgaben). Der Resolutionsbeweis befindet sich zwischen den Zeilen mit den Worten `PROOF` und `end of proof`, im vorliegenden Fall ist er wie folgt:

```
---------------- PROOF ----------------

1 [] m(a).
2 [] -p(a).
3 [] -m(Y)|s(Y).
4 [] -s(Z)|p(Z).
5 [binary,3.1,1.1] s(a).
6 [binary,4.1,5.1] p(a).
7 [binary,6.1,2.1] $F.

------------ end of proof -------------
```

Die ersten vier Zeilen geben die Klauseln der Formel wieder. Die fünfte bis siebte Zeile sind Resolutionsschritte, dabei sind die verwendeten Klauseln in links in eckigen Klammern und rechts der Resolvent angegeben. In der fünften Zeile werden $\{\neg M(y), S(y)\}$ und $\{M(a)\}$ (d.h. die dritte und erste Zeile) zu $\{S(a)\}$ resolviert (die Substitutionen werden nicht explizit angegeben, im vorliegenden Fall ist offensichtlich $s_1 = s_2 = []$ und $sub = [y/a]$.

In der sechsten Zeile wird dann aus $\{\neg S(z), P(z)\}$ und $\{S(a)\}$ (d.h. der vierten und fünften Zeile) die Klausel $\{P(a)\}$ hergeleitet, und in der siebten Zeile dann der Widerspruch.
Gibt es keinen Resolutionsbeweis, so erhält man die Ausgabe

```
Search stopped because sos empty.
```