Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Alexander Krauss, Sascha Böhme

WS 2008/09
30. 01. 2009

Logic

Mini-Project 3

Deadline: Monday, 23 February 2009, 8:00 a.m.
Submit to: `boehmes@in.tum.de`

Discrete Linear Orders

The theory of *discrete linear orders* is defined on the language with a single unary function symbol s ("successor"), and the relation symbols $=$ and $<$. It is given by the following axioms, in addition to the usual axioms that describe equality:

$$\forall xy.\, x = y \vee x < y \vee y < x \qquad \text{(total)}$$
$$\forall xyz.\, x < y \wedge y < z \Rightarrow x < z \qquad \text{(transitive)}$$
$$\forall x.\, \neg(x < x) \qquad \text{(irreflexive)}$$
$$\forall xy.\, x < y \Leftrightarrow s(x) = y \vee s(x) < y \qquad \text{(successor)}$$
$$\forall x.\, \exists y.\, x = s(y) \qquad \text{(predecessor)}$$

As usual, $x \leq y$ can be treated as an abbreviation for $x < y \vee x = y$.

Your task is to find a quantifier elimination procedure for this theory, which is similar to that of dense linear orders.

Since our language is a sub-language of Presburger arithmetic (when writing $x+1$ for $s(x)$), the theory is obviously decidable. However, the quantifier elimination procedure returns a formula that may contain $+$ and divisibility constraints, which is no longer expressible in the original language. However, you can use the decision procedure for Presburger arithmetic to check if the results of your quantifier elimination are correct.

Exercise 3.1

Describe a quantifier elimination for discrete linear orders. Again, it is sufficient to consider the interesting case $\exists x.\, p_1 \wedge p_n$ where $p_1, \ldots, p_n$ are atomic formulas that all contain x.

Exercise 3.2

Implement your algorithm in OCaml, as a function

qelim_dilo : fol formula $\rightarrow$ *fol formula*

Specify soundness and completeness for the function and implement appropriate tests. (Make sure that you have specified all preconditions.) You find some test cases below:

Some test cases

- $\exists y.x < y \wedge y < z$
- $\forall x.x < y \vee y < s(x)$
- $\forall y.y < s(s(x)) \Rightarrow z < s(y) \Rightarrow s(z) \leq s(x)$
- $\neg(\forall zw.w \leq s(z) \vee s(s(z)) \leq w \vee w \leq x \vee x \leq z \vee w \leq y \vee y \leq z)$

Coding and Documentation Guidelines

- Follow the style of coding that you find in the book. You can also use any pieces of the existing code base, in particular the library.
- Keep your code purely functional. You will not need any imperative features of OCaml.
- Do not do low-level optimizations. Always prefer clarity and conciseness over efficiency.
- Find a suitable way of documenting your implementation. This can be comments in the code, a separate text, or a text that includes code snippets as you find it in the book.
- Keep in mind that extensive comments cannot make up for bad code. On the other hand, clear and readable code can make many comments obsolete.

Recommended setup for this project

```
#use "init.ml";;                    (* load everything *)
```