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

WS 2008/09
17. 10. 2008

Logic

Exercise Sheet 1

Exercise 1.1 Does Wango semantle? – A motivation for logic

Calusians wingle.
Wango is Calusian.
Anyone who wingles semantles.

Does Wango semantle?

a) Can you answer the question? Do you know what it means to *semantle*?

b) Translate the text into the formal language of logic, using the connectives you know from previous courses.

c) Can you imagine a computer answering the question?

Exercise 1.2 Setting up your environment

The following exercises deal with the simple arithmetic expressions defined in the introduction:

```
type expression =
  Var of string
  | Const of int
  | Add of expression * expression
  | Mul of expression * expression
```

You should have a working installation of OCaml, and a copy of the book code[1]. At the OCaml prompt, load the complete book code with

```
#use "init.ml";;
```

Then, to have the right book chapter loaded, do a

```
#use "intro.ml";;
```

You can now enter arithmetic expressions and have them parsed for you:

```
<< 3 + x * (y + 1) >>
```

[1]Note that the code needs to be built first using `make`. If you are running windows, then you might not have the *make* tool, and you can get around this by downloading `http://www4.in.tum.de/~krauss/atp_interactive.ml` and placing it in the code directory.

Exercise 1.3 Manipulating expressions in OCaml

a) Write a function `eval` that evaluates expressions for a given valuation, which is a function mapping variable names to values:

   $eval : (string \to int) \to expression \to int$

b) Write a function *subst* that substitutes another term for a variable:

   $subst : string \to expression \to expression \to expression$

c) Create an extended expression type *mexpression* that adds a constructor *Sub* for subtraction. Also extend your evaluation function *meval* to the new type.

   Now write a function

   $convert : mexpression \to expression$

   that eliminates the new construct expressing it in terms of the other ones. Make sure that

   $eval\ env\ (convert\ exp) = meval\ env\ exp$

   holds.

Exercise 1.4 Canonical Forms

Write code that "multiplies out" an expression by applying distributivity, and then reorders the terms to a canonical form.

You can use this code to check if two expressions are equivalent.