Fachbereich Mathematik und Informatik
Prof. Dr. K. Ostermann

Sebastian Erdweg, seba@informatik
Tillmann Rendel, rendel@informatik

Philipps Universität Marburg

October 15, 2009

# Programming Languages and Types
# Homework Assignment H1

In addition to group exercises, there are compulsory homework assignments which you are to work on in groups of three students. To be admitted to the exam, you have to hand in reasonable solutions for all but two assignments, and you have to present your homework in one of the exercises classes. Please hand in your homework by email to `mailto:pllecture@informatik.uni-marburg.de`. The exercise descriptions are published on `http://www.informatik.uni-marburg.de/~kos/teaching/pl` on Thursday evenings after the lecture, and have to be handed in until Thursday of the following week. The deadline for this assignment is October 22.
Please form groups of three students each and notify us via email to `mailto:pllecture@informatik.uni-marburg.de` about who is in your group as soon as possible.

Submit the solutions to H1.1 to H1.4 in a single .ss file. Those tasks which require prose should be written as comments.

## H1.1 DeBruijn Form

Convert (by hand) the following expression to deBruijn form as described in Sec. 3.5 of the textbook:

```
{with {a 42}
  {with {b {+ a 1}
    {with {c {+ a b}
      {+ a b c}}}}}
```

## H1.2 Alpha Equivalence

Two WAE terms are *alpha-equivalent*, if one term can be transformed into the other one by renaming identifiers, i.e., uniformly substituting a binding instance and all corresponding bound instances of an identifier by a different identifier. For example

```
{with {a 5}
  { with {b 6}
    {+ a b}}}
```

is alpha equivalent to

```
{with {a 5}
  { with {c 6}
    {+ a c}}}
```

please turn the page!

It is not allowed, though, to perform a renaming if unrelated identifiers would accidentially be bound, e.g., renaming "b" to "a" in the example above would incorrectly yield

```
{with {a 5}
  {with {a 6}
    {+ a a}}}
```

Implement a function

```
alphaequivalent : WAE, WAE -> Bool
```

which checks whether two WAE expressions are alpha-equivalent. Add a few interesting test cases.

## H1.3 Conversion to DeBruijn Form

Consider again the presentation of deBruijn indices in Sec. 3.5 of the textbook. Here is a data-type to store WAE expressions in deBruijn form:

```
(define-type WAE-dB
  [num-dB (n number?)]
  [add-dB (lhs WAE-dB?) (rhs WAE-dB?)]
  [sub-dB (lhs WAE-dB?) (rhs WAE-dB?)]
  [with-dB (named-expr WAE-dB?) (body WAE-dB?)]
  [id-dB (name number?)])
```

Now define a function

```
to-deBruijn : WAE -> WAE-dB
```

which converts a WAE expression into its equivalent WAE-dB expression. Write at least one test case, including one that uses the example from H1.1.

## H1.4 Alpha Equivalence 2

Think about what alpha-equivalent WAE expressions look like after their conversion to WAE-dB. Now write a function

```
alphaequivalent2 : WAE, WAE -> Bool
```

which has the same meaning as the function in H1.2, but which is implemented using the deBruijn conversion from H1.3. Add a few tests (using the `test` function) to make sure that the two functions are equivalent.

Hint: You may want to use the `equal?` function, which checks recursively for equality of structured data types.