Technische Universität München *Dr. K. N. Verma*
Fakultät für Informatik *verma@in.tum.de*
Prof. Dr. H. Seidl *Room: MI 02.07.041*

Virtual Machines

Summer Semester 2007

Exercise sheet 10 *Deadline: 17 July 2007 12:00*

Aufgabe 1: *Unification: Occur check* *2 Punkte*

Unify t and s and compare the costs for unification **with** and **without** occur check.

$$t \equiv p(X_1, \ldots, X_n)$$
$$s \equiv p(f(X_0, X_0), f(X_1, X_1), \ldots, f(X_{n-1}, X_{n-1}))$$

Aufgabe 2: *Code Generation for WiM* *10 Punkte*

Consider the following program P:

$rev1([], R, R)$:- .
$rev1([X|L], R, A)$:- $rev1(L, [X|R], A)$.
$reverse(L, R)$:- $rev1(L, [], R)$.

?- $reverse(X, [4, 2, 1])$.

(a) Translate P to WiM code (**with** optimization) and point them out.
(b) Execute the WiM code showing the sequence of (sub-)goals that are called and monitor how stack and heap develop after each of these goals has been processed.

Aufgabe 3: *Prolog-arithmetics* *18 Punkte*

In Prolog the arithmetic operators `+`, `-`, `*` and `/` are interpreted as common term constructors. In order to implement an efficient arithmetics, Prolog is extended by the type Integer and the predicate *is/2*. Thus, the Goal "X *is* t" arithmetically evaluates the term t and unifies the result of the evaluation with X.

a) The WiM should be extended by the data type **Integer** and (analogously to atoms) by the abstract instructions for unification of Integers.
b) Define an abstract instruction *eval*, which pops the term t, pointing to the topmost stack cell, from the stack, arithmetically evaluates the term and pushes the result on the stack. If t contains unbounded variables, atoms or structures not built with `+`, `-`, `*` or `/`, the instruction *eval* should fail and raise an error.
c) Analogously to MaMa define abstract instructions for performing arithmetical operations on the stack. Specify a code generation function `code_I` to handle arithmetical expressions on the right-hand side of an *is*-Goal.
d) Specify a code scheme for goals of the form "X *is* t".
e) Generate `code_PR` for the following predicate:

$len([], 6-2*3)$:- .
$len([X|R], L)$:- $len(R, L1), L$ *is* $L1 + 1$.