Formally Specified Computer Algebra Software

Muhammad Taimoor Khan (DK 10)
Supervisor: Prof. Wolfgang Schreiner

Linz March 25, 2011

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

supported by

LAND OBERÖSTERREICH

FWF
Der Wissenschaftsfonds.

JOHANNES KEPLER UNIVERSITÄT LINZ | JKU

Introduction

- Project goals
    - Find errors in computer algebra programs without execution, i.e. by static analysis
    - Programs written in untyped computer algebra languages, i.e. Maple and Mathematica
    - Program annotated with formal specification
        - Types and pre/post-conditions of a method
    - Develop a tool to find errors/inconsistencies in the annotated program
        - Type inconsistencies and violations of method preconditions
- Project results so far
    - Defined and implemented a computer algebra type system (Mar. 2010 - Jan. 2011)
        - Started PhD in October 2009
    - Defining a formal specification language for computer algebra language (Feb. 2011 - Date)

A computer algebra program type checked

```
status:=0;
prod := proc(l::list(Or(integer,float)))::[integer,float];
          global status;
          local i, x::Or(integer,float), si::integer:=1, sf::float:=1.0;
          # π={..., status:anything, i:anything, x:Or(integer,float), si:integer, sf:float}
          for i from 1 by 1 to nops(l) do
              x:=l[i]; status:=i;
              # π={..., i:integer, x:Or(integer,float),..., status:integer}
              if type(x,integer) then
              # π={..., i:integer, x:integer, si:integer, ..., status:integer}
                   if (x = 0) then return [si,sf]; end if; si:=si*x;
              elif type(x,float) then
              # π={..., i:integer, x:float, ..., sf:float, status:integer}
                   if (x < 0.5) then return [si,sf]; end if; sf:=sf*x;
              end if;
              # π={..., i:integer, x:Or(integer,float), si:integer, sf:float, status:integer}
          end do;
          # π={..., status:anything, i:anything, x:Or(integer,float), si:integer, sf:float}
          status:=-1; return [si,sf];
          end proc;
          ...
```

A computer algebra procedure formally specified

```
(*@ requires true;
  @ global status;
  @ ensures (status = -1 and RESULT[1] = mul(e, e in l, type(e,integer))
  @             and RESULT[2] = mul(e, e in l, type(e,float))
  @             and forall(i::integer, 1<=i and i<=nops(l) and type(l[i],integer)
  @                                implies l[i]<>0)
  @             and forall(i::integer, 1<=i and i<=nops(l) and type(l[i],float)
  @                                implies l[i]>=0.5))
  @          or (1<=status and status<=nops(l)
  @             and RESULT[1] = mul(l[i], i=1..status-1, type(l[i],integer))
  @             and RESULT[2] = mul(l[i], i=1..status-1, type(l[i],float))
  @             and ((type(l[status],integer) and l[status]=0)
  @                     or (type(l[status],float) and l[status]<0.5))
  @             and forall(i::integer, 1<=i and i<status and type(l[i],integer)
  @                                implies l[i]<>0)
  @             and forall(i::integer, 1<=i and i<status and type(l[i],float)
  @                                implies l[i]>=0.5));
  @*)

proc(l::list(Or(integer,float)))::[integer,float]; ... end proc;
```

Current status and activities (Mar. 2010 to Date)

- Achievements
  - Defined a substantial subset of Maple called *MiniMaple*
    - EBNF grammar for *MiniMaple*
  - Defined a type system for *MiniMaple*
    - As a simple decidable logic
  - Implemented a type checker for *MiniMaple* in Java
    - Tested with small *MiniMaple* programs

    "M.T.Khan, *A Type Checker for MiniMaple*, Technical Report, RISC, JKU, Linz, March 2011 (to appear)"
- Ongoing research
  - Working on a formal specification language for *MiniMaple* (Feb. 2011 to Date)