Glinda

A Meta-Circular Interpreter for Oz

A Short Presentation

Martin Henz and Michael Mehl

Programming Systems Lab
German Research Center for AI (DFKI)
{henz,mehl}@dfki.uni-sb.de
Saarbrücken, Germany

Execution of Programs

Meta-circular Interpreter for Oz

interprete Oz source program directly
by Oz interpreter program

Ground Term Representation of Programs

```
local Y Square in
   proc {Square X Y}
      {`*` X X Y}
   end
   {Square 4 Y}
   {Browse Y}
end
```

is encoded by

```
Program = loc([´Y´ ´Square´]
              comp(def(´Square´
                       [´X´ ´Y´]
                       appl(´`*`´ [´X´ ´X´ ´Y´]))
                   comp(appl(´Square´ [4 ´Y´])
                        appl(´Browse´ [´Y´]))))
```

A Simple Interpreter

```
proc {Interprete P Env}
   case P
   of loc(VarList Body)
   then {Interprete Body {Extend VarList Env}}
   [] appl(Functor Args)
   then {{Lookup Env Functor} {Lookups Env Args}}
   [] def(Functor Args Body)
   then {Lookup Env Functor
         proc {$ Xs}
            NewEnv={ExtendWithValues Args Xs Env}
         in
            {Interprete Body NewEnv}
         end}
   [] comp(P1 P2)
   then {Interprete P1 Env} {Interprete P2 Env}
end end
```

More Realistic

add:

- **conditionals**
- **disjunctions**
- **values**

Glinda

- compiler/interpreter compatibility
- import/export of any Oz value
- modular structure for easy extension

Potential Applications

- **Profiling/Debugging**
- **Experimentation with Computation Model**
- **teaching programming language semantics**