Prolog Project Notes

- **How to put the Warren compiler together?**

compiler(+SourcefileAsString, ?AbsoluteObjCode) :-

tokenize(?TokenList,+SourcefileAsString, ?Z),

parse(+TokenList, ?AbstractSyntaxTree),

encodestatement(+AbstractSyntaxTree, ?Dict, ?Code),

assembly(+Code,+?Dict, ?AbsoluteObjCode).

Where + indicates an input to the subgoal and ? an output of the subgoal.

Your job is to change this setup so that type checking instead of code generation is the job of the compiler. It may be easiest to return the value of the type checked AST as the result of your program and write out any type errors during the computation.

Prolog Project

An example of compilation of a small program:

compiler(+SourcefileAsString, ?AbsoluteObjCode) :-

compiler("pos = init+ 60;",Code):-

tokenize(?TokenList,+SourcefileAsString, ?Z),

tokenize returns [pos, =, init, +, 60, ;]*//all are atoms except 60 which is a Prolog integer*

parse(+TokenList, ?AbstractSyntaxTree),

parse returns assign(name(pos), expr(+,name(init),const(60))) as AST. During the parse, lookup(init,D,Addr1) and lookup(pos,D,Addr2) are executed so

encodestatement(+AbstractSyntaxTree, ?Dict, ?Code),

the code generator returns:

(instr(load,Addr1), instr(addc, 60), instr(store,Addr2))

assembly(+Code,+?Dict, ?AbsoluteObjCode).

the assembly returns:

(instr(load,5);instr(addc,60); instr(store 6); instr(halt,_); block(2))

Unanswered Questions

- **Is there an easy way to use a file as input for tokenizer?**
- **How will the interleaving of writes with other predicates affect control flow in my program?**
- **How to indicate that the program suffered at least one type error?**
- **How does coercion complicate the type checking? Where is coercion allowed?**
- **What overall strategy to employ: interleaved type checking or separate pass type checking?**
- **What to do about syntax errors?**
- **Are there more????**