CS 603: Programming Language Organization

Lecture 2

Spring 2005

Department of Computer Science

University of Alabama

Joel Jones

Outline

- Questions
- Impcore Examples
- Operational Semantics
- Reading for next time

LISP-like Syntax

```
toplevel ::= exp
           | (use file-name)
           | (val variable-name exp)
           | (define function-name
               (formals) exp)
exp      ::= value
           | variable-name
           | (set variable-name exp)
           | (if exp exp exp)
           | (while exp exp)
           | (begin exp exp)
           | (function {exp})
formals  ::= {variable-name}
value    ::= integer
function ::= function-name
           | primitive
primitive ::= + | - | * | / | = | < | > |
            | print
```

```
integer ::= sequence of digits, possibly
            prefixed with a plus or minus
            sign
*-name    ::= sequence of characters not
              an integer and not
              containing (, ), ;, or
              whitespace
```

Impcore Examples

- `(mult m n)` - multiply m by n without using "*"
- `(sumsquares n)` - 1^2 + 2^2 +…+ n^2

Pair Up:

- Put up solution to mult and sumsquares on the board

Environments

- Set of mappings from names to values (or meanings)
- Operations on environments
  - Lookup–given name, return value: $\rho(x)$

    ρ - environment

    x - name
  - Extend–given name and value, add to mappings: $\rho\{x \mapsto v\}$
    - v - value

Specifying Meaning

- Operational Semantics
  - Mapping from AST (or other abstraction representation) to meaning, in terms of primitives
  - Mechanics involve state and transitions from state to state, involving inference, environments, sets, etc. which define a virtual machine
- Interpreters
  - Given operational semantics, interpreters can be derived, almost (but not quite!) mechanically

Judgements and Rules of Inference

- Transition rules of virtual machine are written in the form of *judgments*
- A judgment is a relation, not a function, which implies that non-deterministic evaluations are possible.
- A judgment consists of premises and a conclusion
- A judgment holds only if all of the premises are true

Operational Semantics of Impcore (State)

- Four parts to state
  - *Toplevel t* or *expression e* being evaluated (matches on AST tags)
  - Value environment holding values of global variables, ξ
  - Function definition environment, ϕ
  - Value environment holding formal parameters, ρ

Operational Semantics of Impcore (Judgements)

- State when evaluating judgments for *toplevel*: ‹*t*, ξ, ϕ›
- State when evaluating judgments for expression: ‹*e*, ξ, ϕ, ρ›
- State between evaluations of *toplevel*:

Pair Up:

- How many elements to state tuple?
- What contents? Why?

- ‹ξ, ϕ›