Programming Languages

2nd edition

Tucker and Noonan

Chapter 8
Semantic Interpretation

To understand a program you must become both the machine and the program.

A. Perlis

Contents

8.1 State Transformations and Partial Functions

8.2 Semantics of Clite

8.3 Semantics with Dynamic Typing

8.4 A Formal Treatment of Semantics

Dynamically Typed Languages

Scripting: Perl, Python, PHP

Object-oriented: Smalltalk, Ruby

Functional: Scheme, ML, Haskel

Logic: Prolog

Our example: dynamically typed C++Lite

```
int main( ) {
n =3; i =1; f =  1.0;
while (i < n) {
    i = i + 1;
    f = f * float(i);
}
}
```

Step	Stmt	n	i	f
1	3	--	--	--
2	4	3	--	--
3	5	3	1	--
4	6	3	1	1.0
5	7	3	1	1.0
6	8	3	2	1.0
7	6	3	2	2.0
8	7	3	2	2.0

Step	Stmt	n	i	f
9	8	3	3	2.0
10	6	3	3	6.0
11	10	3	3	6.0

Perl vs. Python

Perl: implicit conversions, distinct operators

- *“2” < “10” : true – numeric comparison*
- *“2” lt “10” : false – string comparison*
- *2 lt “10” : false – 2 converted to “2”*

Python: explicit conversions required

- *“2” < “10” : false – string comparison*
- *2 < “10” : error*

Meaning Rule 8.10

The meaning of a Program is the meaning of its body when given an empty initial state.

- *Variables declared as encountered*
- *Type of a variable is type of is value*
- *In factorial:*
  - i, n – int
  - f – float

C++Dynamic

Statement = Skip | Block | Assignment | Conditional |

Loop

- *Skip*, *Block unchanged*
- *Conditional*, *Loop – check that* test *is bool*
- *Assignment*
  - add *target* variable to state, if needed
  - no assignment compatibility check needed
  - ???

Meaning Rule 8.11

The meaning of an expression in the current state is a value defied as follows:

- *If the expression is a value, then the value itself*
- *If the expression is a Variable:*
  - If the Variable occurs in the current state, then its associated value.
  - Otherwise the program is meaningless

Meaning Rule 8.11

- *If the expression is a binary:*
  - Determine the value of term1, term2 in current state
  - Apply Rule 4.12 to the operator and values
- *If the expression is a unary:*
  - Determine the value of term in current state
  - Apply Rule 4.13 to the operator and value

See dynamic-expr.java

Meaning Rule 8.12

The meaning of a Binary Expression is a Value:

If operator is arithmetic:

- *If either operand is an int, both operands must be int; perform int addition for +, int subtraction for -, etc.*
- *If either operand is a float, both operands must be float; perform float addition for +, float subtraction for -, etc.*

...

```
Value M (Expression e, State sigma) {
    if (e instanceof Value)
        return (Value)e;
    if (e instanceof Variable) {
        StaticTypeCheck.check( sigma.containsKey(e),
            "reference to undefined variable");
        return (Value)(sigma.get(e));
    }
if (e instanceof Binary) {
        Binary b = (Binary)e;
        return applyBinary (b.op,
                    M(b.term1, sigma), M(b.term2, sigma));
    }
    if (e instanceof Unary) {
        Unary u = (Unary)e;
        return applyUnary(u.op, M(u.term, sigma));
    }
    throw new IllegalArgumentException(
                "should never reach here");
}
```

```
Value applyBinary (Operator op, Value v1, Value v2)
     {
    StaticTypeCheck.check( v1.type( ) == v2.type( ),
                            "mismatched types");
    if (op.ArithmeticOp( )) {
        if (v1.type( ) == Type.INT) {
            if (op.val.equals(Operator.PLUS))
                return new IntValue(
                        v1.intValue( ) + v2.intValue( ));
            if (op.val.equals(Operator.MINUS))
                return new IntValue(
                        v1.intValue( ) - v2.intValue( ));
            ...
```