Concepts of Programming Languages

Lecture 12 - Assignment Semantics

Patrick Donnelly

Montana State University

Spring 2014

Administrivia

Programming #2 : due 03.21
Homework #3 : due 03.31

Reading:

Chapter 7

Ishmael: Surely all this is not without meaning.

Moby Dick by Herman Melville

Assignment Statements

The general syntax:

```
    <target_var> <assign_operator> <expression>
```

The assignment operator:

- = Fortran, BASIC, the C-based languages
- := Ada

= can be bad when it is overloaded for the relational operator for equality (that's why the C-based languages use == as the relational operator)

Conditional Targets

Conditional targets (Perl):

```
($flag ? $total : $subtotal) = 0
```

which is equivalent to

```
if ($flag){
    $total = 0
} else {
    $subtotal = 0
}
```

Compound Assignment Operators

A shorthand method of specifying a commonly needed form of assignment.

Introduced in ALGOL; adopted by C and the C-based languaes

Example

```
a = a + b
```

can be written as

```
a += b
```

Unary Assignment Operators

Unary assignment operators in C-based languages combine increment and decrement operations with assignment

Example

`sum = ++count`	(count incremented, then assigned to sum)
`sum = count++`	(count assigned to sum, then incremented
`count++`	(count incremented)
`-count++`	s(count incremented then negated)

Assignment Semantics

Issues:

1. Assignment statement vs. expression
2. Multiple assignment
3. Copy vs. reference semantics

Assignment as an Expression

In the C-based languages, Perl, and JavaScript, the assignment statement produces a result and can be used as an operand

```
while ((ch = getchar())!= EOF){...}
```

`ch = getchar()` is carried out; the result (assigned to ch) is used as a conditional value for the while statement

Disadvantage: another kind of expression side effect.

Assignment Statement vs. Expression

In most languages, assignment is a statement; cannot appear in an expression.

In C-like languages, assignment is an expression.

Example

- `if (a = 0)` ... // an error
- `while (*p++ = *q++) ;` // strcpy
- `while (ch = getc(fp))` ... // ???
- `while (p = p->next)` ... // ???

Multiple Assignments

Perl, Ruby, and Lua allow multiple-target multiple-source assignments

```
($first, $second, $third) = (20, 30, 40);
```

Also, the following is legal and performs an interchange

```
($first, $second) = ($second, $first);
```

Multiple Assignment

Example

```
a = b = c = 0;
```

Sets all 3 variables to zero.

Do you see any problems with this???

Copy vs. Reference Semantics

Example

```
a = b;
```

Copy:

- `a, b` have same value.
- Changes to either have no effect on other.
- Used in imperative languages.

Reference:

- `a,b` point to the same object.
- A change in object state affects both
- Used by many object-oriented languages.

Example

```
public void add (Object word, Object number) {
    Vector set = (Vector) dict.get(word);
    if (set == null) {  // not in Concordance
        set = new Vector( );
        dict.put(word, set);
    }
    if (allowDupl || !set.contains(number))
        set.addElement(number);
}
```

Assignment in Functional Languages

Identifiers in functional languages are only names of values

ML:

- Names are bound to values with val

```
val fruit = apples + oranges;
```

  If another val for fruit follows, it is a new and different name

F#:

- F#'s let is like ML's val, except let also creates a new scope

Mixed-Mode Assignment

Assignment statements can also be mixed-mode

In Fortran, C, Perl, and C++, any numeric type value can be assigned to any numeric type variable

In Java and C#, only widening assignment coercions are done

In Ada, there is no assignment coercion