Correction of last lecture

Posted to mailing list and webpage

In general, labels are only “jump targets”

Otherwise ignored in computation

L1: s1
L2: s2

If s1 and s2 are “jump free”, and computation begins at L1

s1 computed, then s2
Consider:

```plaintext
switch (1)
{
    case 0: {x = 1; break;}
    case 1: {x = 2; break;}
    default: {x = 3;}
}
```

Translation:

```plaintext
L:
    jump 1, {0=L1; 1=L2 }, D
L1:
    move x, 1
L2:
    move x, 2
D:
    move x, 3
```

Consider:

```plaintext
switch (1)
{
    case 0: {x = 1;}
    case 1: {x = 2;}
    default: {x = 3;}
}
```

Translation:

```plaintext
L:
    jump 1, {0=L1; 1=L2 }, D
L1:
    move x, 1
    jump 1, { }, B
L2:
    move x, 2
    jump 1, { }, B
D:
    move x, 3
    B:
```
SwitchStmt.translate already implemented
BreakStmt.translate must be completed
Keep in mind that...

translate on statement
takes 3 args
  exit_label where return branches
  allocated by FunctionDef
  cont_label where continue branches
  may be null, from loop
  break_label where break branches
  may be null
  from enclosing switch/loop

Also, two constructors for IRJump:

One fine-grained constructor
  jump e, (v1=L1,..,vn=Ln), D

One simple constructor
  jump 1, {}, D
Just as type-checking can be done at:

- Parse-time, folded into parsing
- Post-parsing, as separate pass

IR translation can also be done at parse-time, or post parsing.

Our implementation: post parsing

Parse-time translation more efficient

However, a problem arises.

Parsing implemented as recursive descent into statements

Implies that e.g. breaks parsed before enclosing switches/loops

Problem is that break label not defined upon translation of break statement

That point in code undefined until enclosing loop/switch translated

Not insurmountable; solved with technique called *backpatching*
Details of algorithm in Section 8.6 of Dragon book

High-level description:

Specific labels for jumps unknown at jump construction; set to label placeholder

Create list of partially completed jumps within control structure
Eventually (when enclosing control structure is translated), you know label

```
cjump
jump

?=label X

{ cjump(F), jump }
```

when you know label, complete jumps in list by replacing placeholder with label

```
cjump
jump

?=label X

{ cjump(F), jump }
```

when you know label “finish” the jumps

```
cjump
jump

?=label X

{ cjump(F), jump }
```
Next topic:

Enjoy your vacation.