Foundations of Programming Languages

Control Structures

Prof. Dr. Christoph Reichenbach

Fachbereich 12 / Institut für Informatik

22. Oktober 2014

Conditional statements

- Related to conditional expression, but chooses between *statements*
- Multiple possible paths:
  - Picks exactly one
- Conditionals (yes-or-no)
- Multiple selection
- Pattern matching

Conditionals

C family languages

```
if (a > 0)
    print("greater zero");
else
    print("less than or equal to zero");
```

Python

```
if a > 0:
    print 'greater zero';
elif a == 0:
    print 'equal to zero';
else:
    print 'less than zero';
```

Nested Conditionals

C-family language

```
if (a > 0)
    if (a == 1)
        print(1);
  else print(0);
```

When do we print `0`?

- ‘Dangling Else Problem’
- Instance of ambiguity in language grammar:

$$\begin{array}{lcl} \textit{stmt} & ::= & \text{‘}\texttt{if}\text{’}\langle \textit{bool-expr}\rangle\langle \textit{stmt}\rangle\text{‘}\texttt{else}\text{’}\langle \textit{stmt}\rangle \\ & \mid & \text{‘}\texttt{if}\text{’}\langle \textit{bool-expr}\rangle\langle \textit{stmt}\rangle \end{array}$$

$$\text{‘}\texttt{if}\text{’}\ \langle \textit{bool-expr}\rangle\ \langle \textit{stmt}\rangle\ \text{‘}\texttt{else}\text{’}\ \boxed{\text{‘}\texttt{if}\text{’}\ \langle \textit{bool-expr}\rangle\ \langle \textit{stmt}\rangle}$$

$$\text{‘}\texttt{if}\text{’}\ \langle \textit{bool-expr}\rangle\ \boxed{\text{‘}\texttt{if}\text{’}\ \langle \textit{bool-expr}\rangle\ \langle \textit{stmt}\rangle}\ \text{‘}\texttt{else}\text{’}\ \langle \textit{stmt}\rangle$$

Multiple Selection

C-family language

```
switch (a) {
    case 0: print("0");   // fall through...
    case 1: print("0 or 1");
        break;
    case 2: print("2")
        break;
    default: print("not in [0-2]");
}
```

- Multi-discrimination
- In C-family languages: end cases via **break**
- **default** as ‘catch-all’
- Implemented via combination of:
  - *Multiple conditional branches* (decision tree)
  - *Jump table*: load $pc from table, indexed by ‘a’

Pattern Matching

Standard ML

```
case (list) of
     []  => print "empty list"
   | [a] => print ("list with one element: " ^ a)
   | _   => print "more than one element in list"
```

- `[]`: literal for ‘empty list’
- `[a]`: list with one element
  - a here is *variable*:
    Bound when the pattern is matched
- _: wildcard, default
- Popular in functional languages
- OCaml, Haskell add *guards*: conditional expressions for each branch

Selection based on any comparable expressions

Summary

- *Control structures* affect choice of next statements
- *Conditionals*: choose one of two sides from boolean expression
- *Multiple Selection*: choose one of many options from
  - Integer
  - String (e.g., in Java)
- *Pattern Matching*: choose one of many patterns over arbitrarily complex data types, may bind variables