Basic Blocks and Flow Graphs

We divide the intermediate code of each procedure into basic blocks. A basic block is a piece of straight line code, i.e. there are no jumps in or out of the middle of a block.

The basic blocks within one procedure are organized as a (control) flow graph, or CFG. A flow-graph has
- basic blocks $B_1 \cdots B_n$ as nodes,
- a directed edge $B_1 \rightarrow B_2$ if control can flow from $B_1$ to $B_2$.
- Special nodes ENTER and EXIT that are the source and sink of the graph.

Inside each basic block can be any of the IRs we’ve seen: tuples, trees, DAGs, etc.
Source Code:
\[
X := 20; \textbf{WHILE } X < 10 \textbf{ DO} \\
X := X-1; A[X] := 10; \\
\textbf{IF } X = 4 \textbf{ THEN } X := X - 2; \textbf{ENDIF}; \\
\textbf{ENDDO}; Y := X + 5;
\]

Intermediate Code:

(1) X := 20 
(2) if X>=10 goto (8) 
(3) X := X-1 
(4) A[X] := 10 
(5) if X<>4 goto (7) 
(6) X := X-2 
(7) goto (2) 
(8) Y := X+5

Constructing Basic Blocks
Assume that the input is a list of tuples. How do we find the beginning and end of each basic block?

1. First determine a set of leaders, the first tuple of basic blocks:
   - The first tuple is a leader.
   - Tuple L is a leader if there is a tuple \texttt{if ... goto L or goto L}.
   - Tuple L is a leader if it immediately follows a tuple \texttt{if ... goto B or goto B}.

2. A basic block consists of a leader and all the following tuples until the next leader.

Basic Blocks...

\[
P := 0; \quad I := 1; \quad (1) \quad P := 0 \quad \Leftarrow \text{(Rule 1.a)}
\]

\[
\text{REPEAT}
\]

\[
P := P + I; \quad (2) \quad I := 1
\]

\[
\text{IF } P > 60 \text{ THEN}
\]

\[
P := 0; \quad (3) \quad P := P + I \quad \Leftarrow \text{(Rule 1.b)}
\]

\[
I := 5 \quad (4) \quad \text{IF } P <= 60 \text{ GOTO (7)}
\]

\[
\text{ENDIF;}
\]

\[
(5) \quad P := 0 \quad \Leftarrow \text{(Rule 1.c)}
\]

\[
(6) \quad I := 5
\]

\[
\text{UNTIL } I > 20;
\]

\[
K := P \times 3 \quad (10) \quad K := P \times 3 \quad \Leftarrow \text{(Rule 1.c)}
\]

Summary

\[
B_1: \quad [(1) \ P:=0; \ (2) \ I:=1]
\]

\[
B_2: \quad [(3) \ P:=P+I;
\quad (4) \text{IF } P<=60 \text{ GOTO } B_4]
\]

\[
B_3: \quad [(5) \ P:=0; \ (6) \ I:=5]
\]

\[
B_4: \quad [(7) \ T1:=I*2; \ (8) \ I:=T1+1;
\quad (9) \text{IF } I<=20 \text{ GOTO } B_2]
\]

\[
B_5: \quad [(10) \ K:=P*3]
\]
Readings and References

- Read Louden:  
  Flow Graphs 475–477
- Or, read the Dragon book:  
  Basic Blocks 528–530  
  Flow Graphs 532–534

Summary

- A Control Flow Graph (CFG) is a graph whose nodes are basic blocks. There is an edge from basic block $B_1$ to $B_2$ if control can flow from $B_1$ to $B_2$.
- Control flows in and out of a CFG through two special nodes $\text{ENTER}$ and $\text{EXIT}$.
- We construct a CFG for each procedure. This representation is used during code generation and optimization.
- Java bytecode is a stack-based IR. It was never intended as an UNCOL, but people have still built compilers for Ada, Scheme and other languages that generate Java bytecode. It is painful.
- Microsoft’s MSIL is the latest UNCOL attempt.

Homework I

Translate the program below into quadruples. Identify beginnings and ends of basic blocks. Build the control flow graph.

```
PROGRAM P;
VAR X : INTEGER; Y : REAL;
BEGIN
  X := 1; Y := 5.5;
  WHILE X < 10 DO
    Y := Y + FLOAT(X);
    X := X + 1;
    IF Y > 10 THEN Y := Y * 2.2; ENDIF;
  ENDDO;
END.
```
Draw the control flow graph for the tuples.

```c
int A[5], x, i, n;
for (i = 1; i <= n; i++) {
    if (i < n) {
        x = A[i];
    } else {
        while (x > 4) {
            x = x*2 + A[i];
        }
    }
    x = x + 5;
}
```

1. $i := 1$
2. IF $i > n$ GOTO (14)
3. IF $i >= n$ GOTO (6)
4. $x := A[i]$
5. GOTO (11)
6. IF $x <= 4$ GOTO (11)
7. $T1 := x * 2$
8. $T2 := A[i]$
9. $x := T1 + T2$
10. GOTO (6)
11. $x := x + 5$
12. $i := i + 1$
13. GOTO (2)