Aufgabe 9: Spracherweiterung: erweiterte while-Schleife

Erweitern Sie die while-Anweisung von MicroJava so, dass eine Schleife mehrere Schleifenköpfe und mehrere Schleifenrümpfe haben kann, zum Beispiel:

```
while (n % 2 == 0)     // loop header 1
   n = n / 2;
|| (n % 3 == 0)        // loop header 2
   n = n - 1;
```

Der erste Schleifenkopf wird mit while eingeleitet, die weiteren mit ||. Die Schleifenbedingungen werden sequenziell geprüft. Falls eine davon true ergibt, wird die dazugehörige Anweisung ausgeführt; nach dieser Anweisung springt das Programm wieder an den Beginn der Schleife zurück. Die Schleife terminiert, wenn keine der Schleifenbedingungen true ist. Die kontextfreie Grammatik dazu sieht wie folgt aus:

```
WhileStat = "while" "("Condition ")" Statement { "||" "(" Condition ")" Statement } .
```

Attributieren Sie diese Grammatik so, dass die richtigen Sprunginstruktionen in MicroJava-Bytecode erzeugt werden. Verwenden Sie dazu die im Buch beschriebenen Hilfsmethoden für die Marken- und Sprungverwaltung (Code.jump, Code.fJump und label.here). Beachten Sie, dass Condition einen Cond-Operanden liefert.

Lösung

```
WhileStat                    (.  Label top = new Label();
                                 Operand x; .)
=  "while"                   (. top.here(); .)
   "(" Condition <↑x> ")"    (.  Code.fJump(x);
                                 x.tLabel.here(); .)
   Statement                 (.  Code.jump(top);
                                 x.fLabel.here(); .)

   {  "||"
      "(" Condition <↑x> ")" (.  Code.fJump(x);
                                 x.tLabel.here(); .)
      Statement              (.  Code.jump(top);
                                 x.fLabel.here(); .)
   } .
```

Condition liefert einen Cond-Operanden. Der anschließende False-Jump führt ans Ende des jeweiligen Schleifenrumpfes, wo entweder die Schleife zu Ende ist oder der nächste Schleifenkopf geprüft wird. Da Condition ein zusammengesetzter boolescher Ausdruck sein kann, kann es unaufgelöste True-Jumps geben. Daher wird nach jedem False-Jump die eventuell existierende Kette der True-Jumps aufgelöst (x.tLabel.here()). Am Ende jedes Schleifenrumpfes wird an den Anfang der Schleife zurückgesprungen.