Übungsblatt 07

Til Mohr, 405959
Andrés Montoya, 405409
Marc Ludevid Wulf, 405401

Aufgabe 4

```
x_3 := x_4 + 0;    // Set x_3 to 0
x_0 := x_4 + 1;    // Set x_0 to 1
// Note 1
LOOP x_2 DO
        x_3 := x_4 + 0;   // Set x_3 to 0
        // Note 2
        LOOP x_1 DO
                LOOP x_0 DO
                        x_3 := x_3 + 1
                END
        END;
        x_0 := x_3 + 0
END
```

Unser Programm berechnet für $x_1^{x_2}$ x_2-mal $1 \cdot x_1 \cdot ... \cdot x_1$.
Hierzu müssen wir also x_2 mal eine Multiplikation $x_0 := x_0 \cdot x_1$ durchführen (`Note 1`).
Dazu initialisieren wir x_0 auf 1.
Die Multiplikation selber müssen wir nochmal in 2 Schleifen aufteilen (`Note 2`): Wir setzten unsere Hilfsvariable x_3 auf 0, addieren dann $x_1 \cdot x_0$-mal eine 1 auf x_3, und speichern dann das Ergebnis in x_0.

Aufgabe 5

a) Unser Programm benutzt Variablen zusätzlich zu den aus `P` und `Q`, die die Programme selber nicht benutzen. Diese Variablen sind eben x_l, eine Hilfsvariable x_m und x_{k+1}, welche immer 0 ist.

```
x_m := x_{k+1} + 1;  // Set x_m to 1
WHILE x_l ≠ 0 DO
        Q;
        x_l := x_{k+1} + 0; // Set x_l to 0
        x_m := x_{k+1} + 0 // Set x_m to 0
END;
WHILE x_m ≠ 0 DO
        P;
        x_l := x_{k+1} + 0; // Set x_l to 0
        x_m := x_{k+1} + 0 // Set x_m to 0
END
```

Unser Programm verwendet die Variable x_m als eine Art Mutex. Haben wir nämlich einmal `Q` ausgeführt, können wir `P` nicht ausführen.

b) Unser Programm benutzt Variablen zusätzlich zu denen aus `P_i`, die die Programme selber nicht benutzen. Diese Variablen sind eben x_{k+1}, eine Hilfsvariable x_m und x_l, welche immer 0 ist.

```
x_{k+1} := x_{k+1} + 1;  // Increase x_{k+1} by 1
x_m := x_l + 1;    // Set x_m to 1

// Loop

WHILE x_{k+1} ≠ 0 DO
        x_{k+1} := x_{k+1} − 1; // Decrease x_{k+1} by 1


        <−−−−>>>    // Weitere Faelle


        WHILE x_m ≠ 0 DO
                P_0;
                x_m := x_l + 0  // Set x_m to 0
        END
END
```

Unser Programm verwendet die Variable x_m als eine Art Mutex. Haben wir nämlich einmal ein Programm `P_i` ausgeführt, so können wir kein anderes Programm mehr ausführen. Hierbei ersetzt man `<---->>>` so häufig mit der Äußeren Schleife (`P_i` anpassen!), wie es Programme gibt. Die Schleife, die quasi $x_{k+1} = n$ überprüft (wobei n der höchste Index der Programme ist) noch eine Sonderschleife bekommt. Beispiel für Programme `P_0, P_1, P_2`:

```
x_{k+1} := x_{k+1} + 1;  // Increase x_{k+1} by 1
x_m := x_l + 1;    // Set x_m to 1

// Loop

WHILE x_{k+1} ≠ 0 DO
        x_{k+1} := x_{k+1} − 1; // Decrease x_{k+1} by 1


        WHILE x_{k+1} ≠ 0 DO
                x_{k+1} := x_{k+1} − 1; // Decrease x_{k+1} by 1


                WHILE x_{k+1} ≠ 0 DO
                        x_{k+1} := x_{k+1} − 1; // Decrease x_{k+1} by 1

                        WHILE x_{k+1} ≠ 0 DO
                                x_m := x_l + 0  // Set x_m to 0
                        END

                        WHILE x_m ≠ 0 DO
                                P_2;
                                x_m := x_l + 0  // Set x_m to 0
                        END
                END
```

```
            WHILE x_m ≠ 0 DO
                P_1;
                x_m := x_l + 0  // Set x_m to 0
            END
        END


        WHILE x_m ≠ 0 DO
            P_0;
            x_m := x_l + 0  // Set x_m to 0
        END
END
```

Til Mohr, 405959
Andrés Montoya, 405409
Marc Ludevid Wulf, 405401

Aufgabe 6

100-VARIABLE-WHILE Programme sind Turing-mächtig, da sie TMs simulieren können. O.B.d.A: Wir betrachten nun eine eine TM mit dem Bandalphabet 0,1,B und k Zuständen wobei der Zustand 0 der Endzustand ist.

Um die TM zu simulieren benötigt ein WHILE Programm folgende Variablen: x_0 als Zustand x_1 als Wort vor dem Lesekopf x_2 als Wort nach dem Lesekopf x_3 als Hilfsvariable für die if-Statements

Das Programm sieht dann folgendermassen aus: Eine äussere While-Schleife durchläuft das Programm solange x_0 nicht 0 (Endzustand) ist. In jedem Schleifendurchlauf wird mit k if-Statements geprüft in welchem Zustand sich die TM befindet und mit einem weiteren if-Statement geprüft was unter dem Lesekopf steht. Dann wird dementsprechend das Band und der Zustand erneuert. Die if-Statements für die Zustände brauchen eine Hilfsvariable (siehe Aufgabe 5.b)) Das Updaten des Zustands geht ohne weitere Hilfsvariablen. Für das Updaten des Bandes wird ebenfalls keine Hilfsvariablen benötigt.

Damit kann eine 100-VARIABLE-WHILE Programm die TM locker simulieren und die 100-VARIABLE-WHILE Programme sind Turing-mächtig.