Technische Universität München
Fakultät für Informatik
Prof. Dr. A. Knoll

SS 2006
Übungsblatt 5
30. Juni 2006

Übungen zu Einführung in die Informatik II

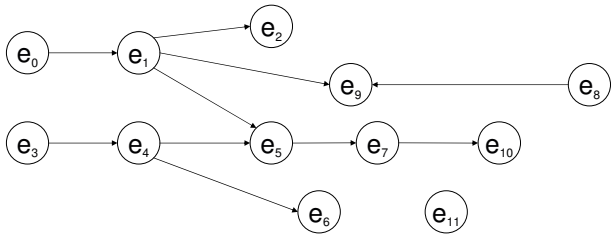
Aufgabe 15 Kleidung

a) Wir definieren zunächst die Aktionenmenge $A = \{a_0, ..., a_n\}$:

a_0 : `Anziehen der Unterhose`
a_1 : `Anziehen der Hose`
a_2 : `Anziehen des Guertels`
a_3 : `Anziehen des Unterhemds`
a_4 : `Anziehen des Hemds`
a_5 : `Anlegen der Hosentraeger`
a_6 : `Anlegen der Krawatte`
a_7 : `Anziehen des Jackets`
a_8 : `Anziehen der Socken`
a_9 : `Anziehen der Schuhe`
a_{10} : `Anlegen des Mantels`
a_{11} : `Aufsetzen des Huts`

Die Definition der Ereignismenge E lässt sich nicht direkt aus realen Gegebenheiten herleiten. Wir nehmen daher an, dass es sich bei einem Ereignis quasi um den Willen der Person handelt, ein bestimmtes Kleidungsstück anlegen zu wohlen. D.h. wir definieren für jede der o.g. Aktionen jeweils ein Ereignis e_i das die Aktion auslöst. Es gilt also $\alpha(e_i) = a_i$
Zur Definition des Prozesses benötigen wir schließlich noch die Kausalitätsrelation $<\subseteq E \times E$, d.h. eine Festlegung welches Ereignis unbedingt vor einem anderen stattfinden muss. Die Elemente der Menge ergeben sich aus der Kausalität obiger Aktionen, die ja mit genau einem Ereignis verbunden sind. So folgt z.B. aus der Tatsache, dass das Unterhemd vor dem Hemd angezogen wird $e_3 < e_4$, bzw. aus der Tatsache, dass man das Jacket erst nach den Hosenträgern anziehen kann $e_5 < e_7$ usw. Da wir $<$ als transitive Hülle definiert haben, reicht es völlig aus nur "benachbarte Kausalitäten" aufzunehmen. Als Definition erhalten wir somit $<\subseteq E \times E$ als transitive Hülle der Menge
$\{(e_3,e_4),(e_5,e_7),(e_4,e_6),(e_4,e_6),(e_8,e_9),(e_0,e_1),(e_1,e_2),(e_1,e_5),(e_9,e_1),(e_7,e_{10}),(e_1,e_{10}),\}$.
Damit haben wir den Prozess $P = (E, <, \alpha)$ vollständig beschrieben.

b) Aus der obigen Definition von < ergibt sich folgendes Aktionsdiagramm:



c) Eine vollständige Sequenzialisierung wäre z.B.:
$e_0 \mapsto e_1 \mapsto e_8 \mapsto e_9 \mapsto e_2 \mapsto e_3 \mapsto e_4 \mapsto e_6 \mapsto e_5 \mapsto e_7 \mapsto e_{11} \mapsto e_{10}$

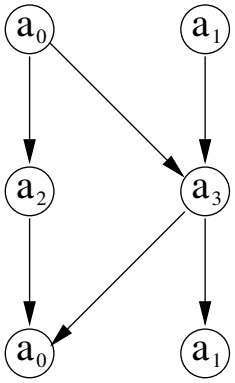
c) Transitivität bedeutet im Fall der Kausalitätsrelation: $(a < b) \wedge (b < c) \Rightarrow (a < c)$, also z.B. folgt aus der Tatsache, dass das Unterhemd vor dem Hemd und dieses wiederum vor dem Jacket angezogen werden muss, dass das Unterhemd vor dem Jacket angezogen werden muss.
Wegen der Transitivität der Relation lässt sich auch leicht zeigen, dass die Unterhose vorm Jacket angezogen werden muss: die Unterhose muss vor der Hose angezogen werden und diese vor den Hosenträgern ($(e_0, e_1) \wedge (e_1, e_5) \Rightarrow (e_0, e_5)$). Das Jacket kann wiederum erst nach den Hosenträgern angezogen werden ($(e_5, e_7) \wedge (e_0, e_5) \Rightarrow (e_0, e_7) q.e.d.$)

Aufgabe 16 Kleinstes Präfix

Ausgehend von der Ereignismenge $E = \{e_i : i \in I\!N_0\}$ und der Aktionenmenge $A = \{a_0, a_1, a_2, a_3\}$ sei der Prozess $P = (E, <, \alpha)$ gegeben durch

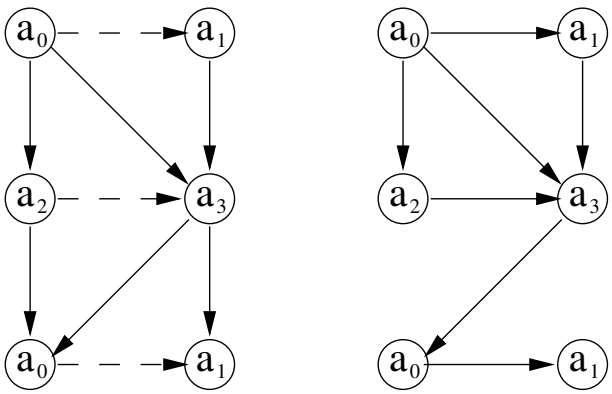
$\alpha : E \to A$ def. durch $\alpha(e_i) = a_{i mod 4}$ für $i \in I\!N_0$
$< \quad \subseteq E \times E$ def. als die transitive Hülle der Menge
$\{(e_i, e_{i+2}), (e_{4i}, e_{4i+3}), (e_{4i+3}, e_{4i+4}) : i \in I\!N_0\}$

a) Ein Präfix von P ist ein Prozess $(E', <', \alpha')$ mit
$E' \subseteq E, \quad <' = < |_{E' \times E'}, \quad \alpha' = \alpha|_{E'}$ und $\forall e \in E, e' \in E' : e < e' \Rightarrow e \in E'$.
Das kleinste Präfix $P_{4,5}$ von P, das die Ereignisse e_4 und e_5 enthält:



$P_{4,5}$

b) Eine unvollständige Sequentialisierung von $P_{4,5}$ ergibt sich z. B. durch Hinzufügen der drei gestrichelten Kanten und Weglassen jeder Kante, zwischen deren Endpunkten es noch einen anderen Pfad gibt. Lässt man eine der drei gestrichelten Kanten aus, ist die Sequentialisierung natürlich auch unvollständig.



Eine vollständige Sequentialisierung erhält man durch *topologisches Sortieren*. Dabei sind die Knoten derart in einer Linie anzuordnen, dass alle Kanten in die gleiche Richtung zeigen. (Das lässt sich leicht bewerkstelligen, indem man mit einem Knoten beginnt, auf den keine Kante zeigt, diesen aus dem Graphen samt seinen ausgehenden Kanten entfernt und so fortfährt, bis die Knotenmenge erschöpft ist. Zyklenfreiheit ist bei Prozessen gegeben.) Die lineare Ordnung, die sich daraus in derselben Richtung zwischen den Knoten ergibt, erweitert die ursprüngliche Relation. Das Ergebnis einer vollständigen Sequentialisierung ist ein sequentieller Prozess.

c) Die Spuren (die Aktionsströme der vollständigen Sequentialisierungen) von $\mathsf{P}_{4,5}$:

$< a_0, a_1, a_2, a_3, a_0, a_1 >$ $\quad$ $< a_1, a_0, a_2, a_3, a_0, a_1 >$
$< a_0, a_1, a_2, a_3, a_1, a_0 >$ $\quad$ $< a_1, a_0, a_2, a_3, a_1, a_0 >$
$< a_0, a_1, a_3, a_2, a_0, a_1 >$ $\quad$ $< a_1, a_0, a_3, a_2, a_0, a_1 >$
$< a_0, a_1, a_3, a_2, a_1, a_0 >$ $\quad$ $< a_1, a_0, a_3, a_2, a_1, a_0 >$
$< a_0, a_1, a_3, a_1, a_2, a_0 >$ $\quad$ $< a_1, a_0, a_3, a_1, a_2, a_0 >$
$< a_0, a_2, a_1, a_3, a_0, a_1 >$
$< a_0, a_2, a_1, a_3, a_1, a_0 >$

Aufgabe 17 Aktionsstruktur (Prozess)

a) Zur Definition der Aktionsstruktur $(E_0, <_0, \alpha)$ über einer Menge E von Ereignissen und einer Menge A von Aktionen gehört die Forderung, dass die Kausalitätsrelation $<_0$ endlich fundiert ist.

(1) Die Forderung drückt aus, dass jedes Ereignis (auch in unendlichen Prozessen) nur endlich viele kausale Vorgänger besitzt. Darauf beschränkt sich unsere Theorie, in der Annahme ein breites Spektrum von Anwendungen zu erfassen.

(2) Eine partielle Ordnung heißt Noethersch, wenn es keine unendlich absteigende Kette gibt. Beispiel einer Noetherschen aber nicht endlich fundierten partiellen Ordnung: Auf den natürlichen Zahlen stehe jede gerade Zahl zu allen ungeraden Zahlen in Relation, und sonst stehe nur jede Zahl zu sich selbst in Relation.

b) Sei $P_1 = (E_1, <_1, \alpha_1)$ der Prozess

$$(\{e_1, e_2\}, \{(e_1, e_1), (e_2, e_2)\}, \{e_1 \mapsto a_1, e_2 \mapsto a_2\})$$

und $P_2 = (E_2, <_2, \alpha_2)$ der Prozess

$$(\{e_1, e_2\}, \{(e_1, e_1), (e_2, e_2)\}, \{e_1 \mapsto a_2, e_2 \mapsto a_1\})$$

(1) P_1 ist <u>kein</u> Teilprozess von P_2,

(2) P_1 ist <u>keine</u> Sequentialisierung von P_2,

(3) P_1 ist aber isomorph zu P_2.

Begründung zu (1) und (2): Die Abbildungen $\alpha_1|E_2$ und α_2 sind verschieden.

Begründung zu (3): Die Abbildung $\varphi : E_1 \to E_2$, die die beiden Elemente vertauscht, ist surjektiv. Die Kausalitätsrelation bleibt erhalten, d.h.

$e_i <_1 e_k$ gilt genau dann, wenn $\varphi(e_i) <_2 \varphi(e_k)$ für $i = 1,2$ und $k = 1,2$

und für die Aktionszuordnungen gilt

$\alpha_1(e_i) = \alpha_2(\varphi(e_i))$ für $i = 1,2$.

Aufgabe 18 Bank-Konto in Java (Lösungsvorschlag)

a) Account.java

```
public class Account {
  int money;

  Account(int money) {
    this.money = money;
  }

  public synchronized void deposit(int money) {
    this.money += money;
    System.out.println(money + "\tdeposited\t\t\t" + this.
       money + "\tremaining");
    notifyAll();
  }

  public synchronized void withdraw(int money) {
    while (this.money < money) {
      try {
        wait();
      } catch (InterruptedException e) {
        break;
      }
    }

    this.money -= money;
    System.out.println("\t\t\t" + money + "\twithdrawn\t" +
       this.money + "\tremaining");
    notifyAll();
  }
}
```

b) Depositer.java

```
public class Depositer extends Thread {
  private Account account;
  private int money;

  Depositer(Account account, int money) {
    this.account = account;
```

```
    this.money = money;
  }

  public void run() {
    while (true) {
      this.account.deposit(this.money);

      try {
        sleep((int) (Math.random() * 1000));
      } catch (InterruptedException e) {
        break;
      }
    }
  }
}
```

c) Withdrawer.java

```
public class Withdrawer extends Thread {
  private Account account;
  private int money;

  Withdrawer(Account account, int money) {
    this.account = account;
    this.money = money;
  }

  public void run() {
    while (true) {
      this.account.withdraw(this.money);

      try {
        sleep((int) (Math.random() * 1000));
      } catch (InterruptedException e) {
        break;
      }
    }
  }
}
```

d) BankAccount.java

```
public class BankAccount {
  public static void main(String[] args) {
    Account account = new Account(0);
    Depositer depositer = new Depositer(account, 40);
    Depositer depositer2 = new Depositer(account, 10);
    Withdrawer withdrawer = new Withdrawer(account, 50);
    Withdrawer withdrawer2 = new Withdrawer(account, 70);
    depositer.start();
    depositer2.start();
    withdrawer.start();
```

```
        withdrawer2.start();
    }
}
```