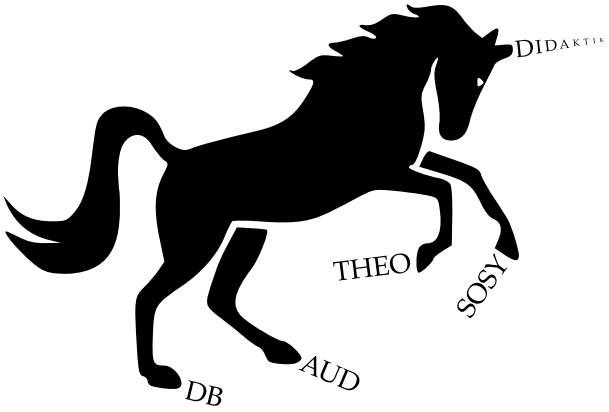
66116
Frühjahr
2017

Datenbanksysteme / Softwaretechnologie (vertieft)

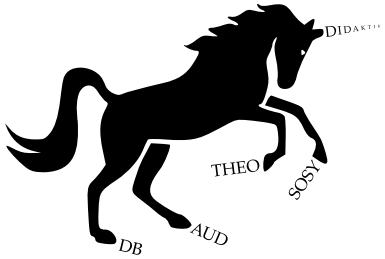
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Teilaufgabe Nr. 1

Aufgabe 2 [Aufbau eines B-Baums]

Konstruieren Sie einen B-Baum, dessen Knoten maximal 4 Einträge enthalten können, indem Sie der Reihe nach diese Suchschlüsssel einfügen:

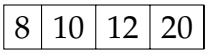
8, 10, 12, 20, 5, 30, 25, 11

Anschließend löschen Sie den Eintrag mit dem Suchschlüssel 8.

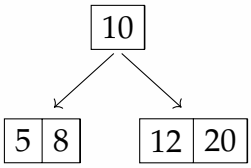
Zeigen Sie jeweils graphisch den entstehenden Baum nach relevanten Zwischenschritten; insbesondere nach Einfügen der 5 sowie nach dem Einfügen der 11 und nach dem Löschen der 8.

Lösungsvorschlag

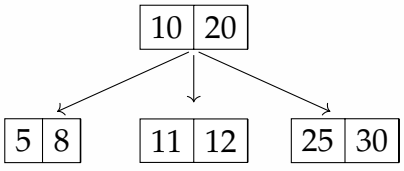
- Schlüsselwert 8 (einfaches Einfügen)
- Schlüsselwert 10 (einfaches Einfügen)
- Schlüsselwert 12 (einfaches Einfügen)
- Schlüsselwert 20 (einfaches Einfügen)



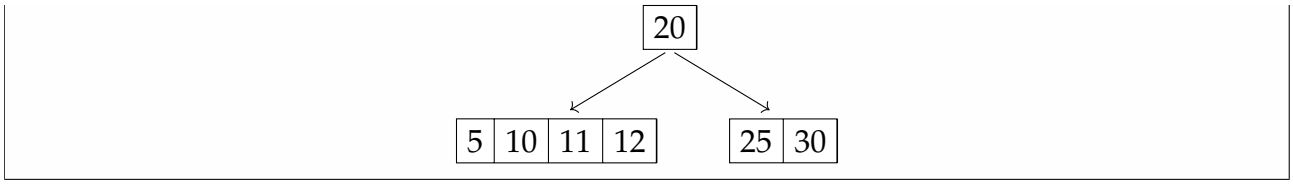
- Schlüsselwert 5 (Split)



- Schlüsselwert 30 (einfaches Einfügen)
- Schlüsselwert 25 (einfaches Einfügen)
- Schlüsselwert 11 (Split)



- Löschen des Schlüsselwerts 8 (Mischen/Verschmelzen)


20
5
10
11
12
25
30

Thema Nr. 2

Teilaufgabe Nr. 1

Aufgabe 5 [Entwurfstheorie]

In der folgenden Datenbank sind die Ausleihvorgänge einer Bibliothek gespeichert:

| Ausleihe | LNr | Name | Adresse | BNr | Titel | Kategorie | ExemplarNr | 1 | Müller | Winklerstr. 1 Datenbanksysteme | Informatik 1 1 | Miller | Winklerstr. 1 | Datenbanksysteme | Informatik 2 2 | Huber | Friedrichstr. | 2 Anatomie I Medizin 5 2 Huber | Friedrichstr. 3 Harry Potter Literatur 20 3 Meier | Bismarkstr. 4 OODBS Informatik 1 4 Meier Marktpl. 5 | Pippi Langstrumpf | Literatur 1

Für die Datenbank gilt:

Jeder Leser hat eine eindeutige Lesernummer (LNr), einen Namen und eine Adresse. Ein Buch hat eine Buchnummer (BNr), einen Titel und eine Kategorie. Es kann mehrere Exemplare eines Buches geben, welche durch eine, innerhalb einer Buchnummer eindeutigen, Exemplarnummer unterschieden werden.

(a) Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können.

(b) Nachfolgend sind alle nicht-trivialen funktionalen Abhängigkeiten, welche in der obigen Datenbank gelten, angegeben:

$$
\mathrm{FA} = \Big\{
\begin{aligned}
\{\, LNr \,\} &\rightarrow \{\, Name \,\},\\
\{\, LNr \,\} &\rightarrow \{\, Adresse \,\},\\
\{\, BNr \,\} &\rightarrow \{\, Titel \,\},\\
\{\, BNr \,\} &\rightarrow \{\, Kategorie \,\},\\
\{\, LNr, BNr, ExemplarNr \,\} &\rightarrow \{\, Name, Adresse, Titel, Kategorie \,\},
\end{aligned}
\Big\}
$$

Einziger Schlüsselkandidat ist { *LNr, BNr, ExemplarNr* }. Überführen Sie das Schema mit Hilfe des Synthesealgorithmus für 3NF in die dritte Normalform.

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— *Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden.* —

i. **Linksreduktion**

— *Führe für jede funktionale Anhängigkeit $\alpha \rightarrow \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$.* —

$$\text{AttrHülle}(FA, \{LNr, BNr, ExemplarNr \setminus LNr\}) = \{Titel, Kategorie\}$$

$$\text{AttrHülle}(FA, \{LNr, BNr, ExemplarNr \setminus BNr\}) = \{Name, Adresse\}$$

$$\text{AttrHülle}(FA, \{LNr, BNr, ExemplarNr \setminus \boldsymbol{ExemplarNr}\}) = \{\boldsymbol{Name, Adresse, Titel, Kategorie}\}$$

$$\text{FA} = \Big\{ \begin{array}{r} \{\, LNr \,\} \to \{\, Name \,\}, \\ \{\, LNr \,\} \to \{\, Adresse \,\}, \\ \{\, BNr \,\} \to \{\, Titel \,\}, \\ \{\, BNr \,\} \to \{\, Kategorie \,\}, \\ \{\, LNr, BNr \,\} \to \{\, Name, Adresse, Titel, Kategorie \,\}, \end{array} \Big\}$$

ii. **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt. ———————————————

$$\text{AttrHülle}(FA - (\{\, LNr \,\} \to \{\, Name \,\}) \cup (\{\, LNr \,\} \to \{\, \varnothing \,\}), \{LNr\}) = \{Adresse\}$$

$$\text{AttrHülle}(FA - (\{\, LNr \,\} \to \{\, Adresse \,\}) \cup (\{\, LNr \,\} \to \{\, \varnothing \,\}), \{LNr\}) = \{Name\}$$

$$\text{AttrHülle}(FA - (\{\, BNr \,\} \to \{\, Titel \,\}) \cup (\{\, BNr \,\} \to \{\, \varnothing \,\}), \{BNr\}) = \{Kategorie\}$$

$$\text{AttrHülle}(FA - (\{\, BNr \,\} \to \{\, Kategorie \,\}) \cup (\{\, BNr \,\} \to \{\, \varnothing \,\}), \{BNr\}) = \{Titel\}$$

$$\text{AttrHülle}(FA - (\{\, LNr, BNr \,\} \to \{\, Name, Adresse, Titel, Kategorie \,\}) \cup (\{\, LNr, BNr \,\} \to \{\, Adresse, Titel, \quad \{Name, Adresse, Titel, Kategorie\}$$

$$\text{AttrHülle}(FA - (\{\, LNr, BNr \,\} \to \{\, Name, Adresse, Titel, Kategorie \,\}) \cup (\{\, LNr, BNr \,\} \to \{\, Name, Titel, K \quad \{Name, Adresse, Titel, Kategorie\}$$

$$\text{AttrHülle}(FA - (\{\, LNr, BNr \,\} \to \{\, Name, Adresse, Titel, Kategorie \,\}) \cup (\{\, LNr, BNr \,\} \to \{\, Name, Adresse \quad \{Name, Adresse, Titel, Kategorie\}$$

$$\text{AttrHülle}(FA - (\{\, LNr, BNr \,\} \to \{\, Name, Adresse, Titel, Kategorie \,\}) \cup (\{\, LNr, BNr \,\} \to \{\, Name, Adresse \quad \{Name, Adresse, Titel, Kategorie\}$$

$$\text{FA} = \Big\{ \begin{aligned} \{\, LNr \,\} &\to \{\, Name \,\}, \\ \{\, LNr \,\} &\to \{\, Adresse \,\}, \\ \{\, BNr \,\} &\to \{\, Titel \,\}, \\ \{\, BNr \,\} &\to \{\, Kategorie \,\}, \\ \{\, LNr, BNr \,\} &\to \{\, \varnothing \,\}, \end{aligned} \Big\}$$

iii. **Löschen leerer Klauseln**

— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* ——

$$\text{FA} = \Big\{ \begin{aligned} \{\, LNr \,\} &\to \{\, Name \,\}, \\ \{\, LNr \,\} &\to \{\, Adresse \,\}, \\ \{\, BNr \,\} &\to \{\, Titel \,\}, \\ \{\, BNr \,\} &\to \{\, Kategorie \,\}, \end{aligned} \Big\}$$

iv. **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* ——

$$\text{FA} = \Big\{ \begin{aligned} \{\, LNr \,\} &\to \{\, Name, Adresse \,\}, \\ \{\, BNr \,\} &\to \{\, Titel, Kategorie \,\}, \end{aligned} \Big\}$$

(ii) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.* ——

(iii) **Schlüssel hinzufügen**

— *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* ——

(iv) **Entfernung überflüssiger Teilschemata**

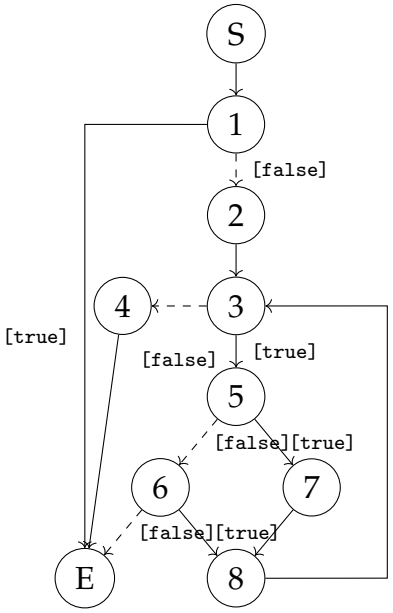
— *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* ——

Teilaufgabe Nr. 2

Aufgabe 1 [Methode „binToInt()“ und Kontrollflussgraph]

Gegeben Sei folgende Methode und ihr Kontrollflussgraph:

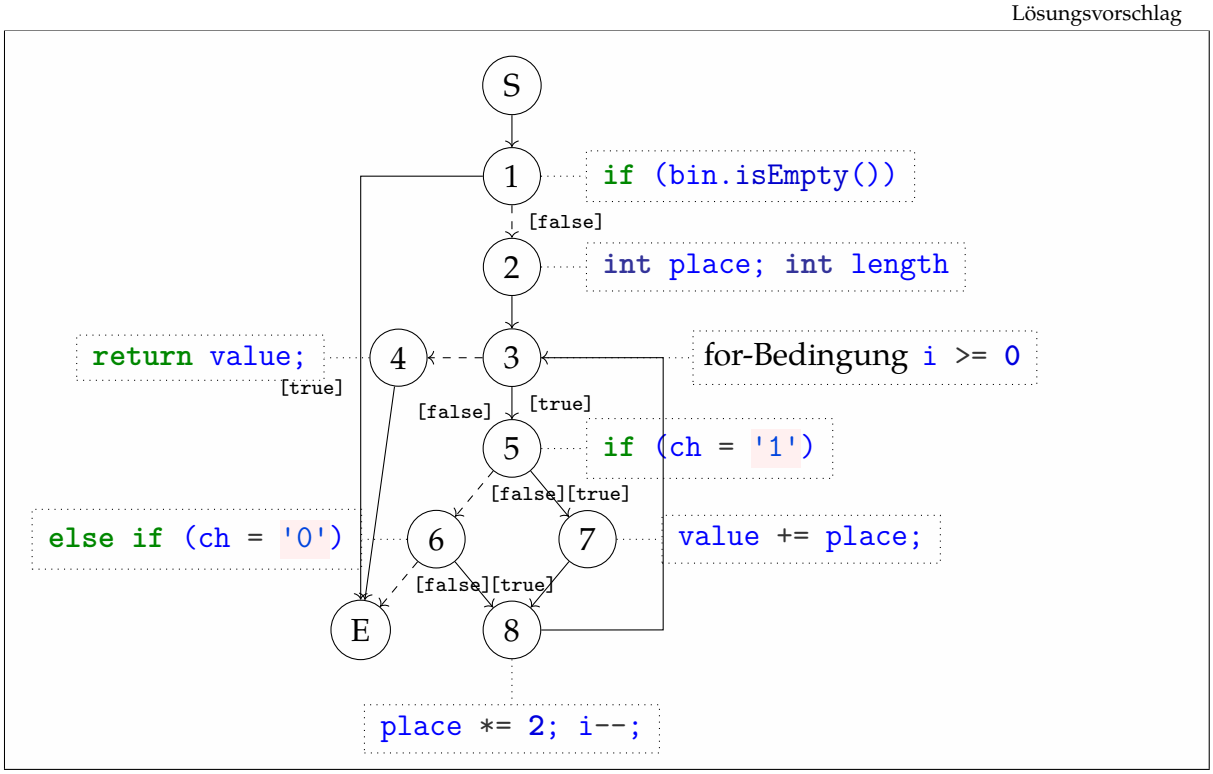
```java
int binToInt(String bin) {
  if (bin.isEmpty())
    return -1;
  int place = 1, value = 0;
  int length = bin.length() -
↪ 1;
  for (int i = length; i >= 0;
↪ --i) {
    char ch = bin.charAt(i);
    if (ch == '1') {
      value += place;
    } else if (ch == '0') {
      // do nothing
    } else {
      return -1;
    }
    place *= 2;
  }
  return value;
}
```



Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/sosy/ab_7/Aufgabe5.java`

(a) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen

Lösungsvorschlag



(i) Verzweigungsüberdeckung

Lösungsvorschlag

p1 (Pfad 1) S 1 E
p2 S 1 2 3 4 E
p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E

(ii) Schleife-Inneres-Überdeckung

Lösungsvorschlag

Äußere Pfade (äußere Pfade): (ohne Ausführung der Wiederholung)

p1 S 1 E
p2 S 1 2 3 4 E

Grenzpfade (boundary test) (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

p4 S 1 2 3 5 6 E

Innere Pfade (interior test) (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)

p5 S 1 2 3 5 7 8 3 5 7 8 3 4 E
p6 S 1 2 3 5 7 8 3 5 6 8 3 4 E
p7 S 1 2 3 5 6 8 3 5 6 8 3 4 E
p8 S 1 2 3 5 6 8 3 5 7 8 3 4 E
p9 S 1 2 3 5 7 8 3 5 7 8 3 5 6 E
p10 = p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E
p11 S 1 2 3 5 6 8 3 5 6 8 3 5 6 E
p12 S 1 2 3 5 6 8 3 5 7 8 3 5 6 E

mit minimaler Testfallanzahl genügen würden.

(b) Welche der vorangehend ermittelten Pfade sind mittels Testfälle tatsächlich überdeckbar („feasible“)? Falls der Pfad ausführbar ist, geben Sie bitte den Testfall an, andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Lösungsvorschlag

Erweitere Methode, die die Knotennamen ausgibt:

```
public static final String RESET = "\u001B[0m";
public static final String ROT = "\u001B[31m";
public static final String GRÜN = "\u001B[32m";

static int binToIntLog(String bin) {
  System.out.println("\nInput: " + bin);
  System.out.print("S");
  System.out.print(1);
  if (bin.isEmpty()) {
    System.out.print("E");
    System.out.println("\nOutput: " + -1);
    return -1;
```

```
    }
    System.out.print(2);
    int place = 1, value = 0;
    int length = bin.length() - 1;
    System.out.print(3);
    for (int i = length; i >= 0; --i) {
      char ch = bin.charAt(i);
      System.out.print(5);
      if (ch == '1') {
        System.out.print(ROT + 7 + RESET);
        value += place;
      } else {
        System.out.print(GRÜN + 6 + RESET);
        if (ch == '0') {
          // do nothing
        } else {
          System.out.print("E");
          System.out.println("\nOutput: " + -1);
          return -1;
        }
      }
      System.out.print(8);
      place *= 2;
      System.out.print(3);
    }
    System.out.print(4);
    System.out.print("E");
    System.out.println("\nOutput: " + value);
    return value;
  }

  public static void main(String[] args) {
    binToIntLog(""); // p1
    binToIntLog("??"); // p2 not feasible
    binToIntLog("x01"); // p3
    binToIntLog("x"); // p4

    binToIntLog("11"); // p5
    binToIntLog("01"); // p6
    binToIntLog("00"); // p7
    binToIntLog("10"); // p8

    binToIntLog("x11"); // p9
    binToIntLog("x01"); // p10
    binToIntLog("x00"); // p11
    binToIntLog("x10"); // p12
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/sosy/ab_7/Aufgabe5.java`

Alle mit Ausnahme von p2.

p2 ist nicht überdeckbar. Passiert ein Wert der Variable `bin` die erste if-Verzweigung,

dann hat der Wert eine Länge größer 0 und betritt deshalb die Wiederholung mit fester Anzahl.

p1	S 1 E	`binToInt("");`
p2	S 1 2 3 4 E	not feasible
p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	`binToInt("x01");`
p4	S 1 2 3 5 6 E	`binToInt("x");`
p5	S 1 2 3 5 7 8 3 5 7 8 3 4 E	`binToInt("11");`
p6	S 1 2 3 5 7 8 3 5 6 8 3 4 E	`binToInt("01");`
p7	S 1 2 3 5 6 8 3 5 6 8 3 4 E	`binToInt("00");`
p8	S 1 2 3 5 6 8 3 5 7 8 3 4 E	`binToInt("10");`
p9	S 1 2 3 5 7 8 3 5 7 8 3 5 6 E	`binToInt("x11");`
p10 = p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	`binToInt("x01");`
p11	S 1 2 3 5 6 8 3 5 6 8 3 5 6 E	`binToInt("x00");`
p12	S 1 2 3 5 6 8 3 5 7 8 3 5 6 E	`binToInt("x10");`

(c) Bestimmen Sie anhand des Kontrollflussgraphen die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach Mc-Cabe.

Lösungsvorschlag

Binärverzweigungen 4

Knoten 10

Kanten 13

Anhand der Binärverzweigungen:

$$\begin{aligned} M &= b + p \\ &= 4 + 1 \\ &= 5 \end{aligned}$$

oder durch Anzahl Kanten e und Knoten n

$$\begin{aligned} M &= e - n + 2p \\ &= 13 - 10 + 2 \cdot 1 \\ &= 5 \end{aligned}$$

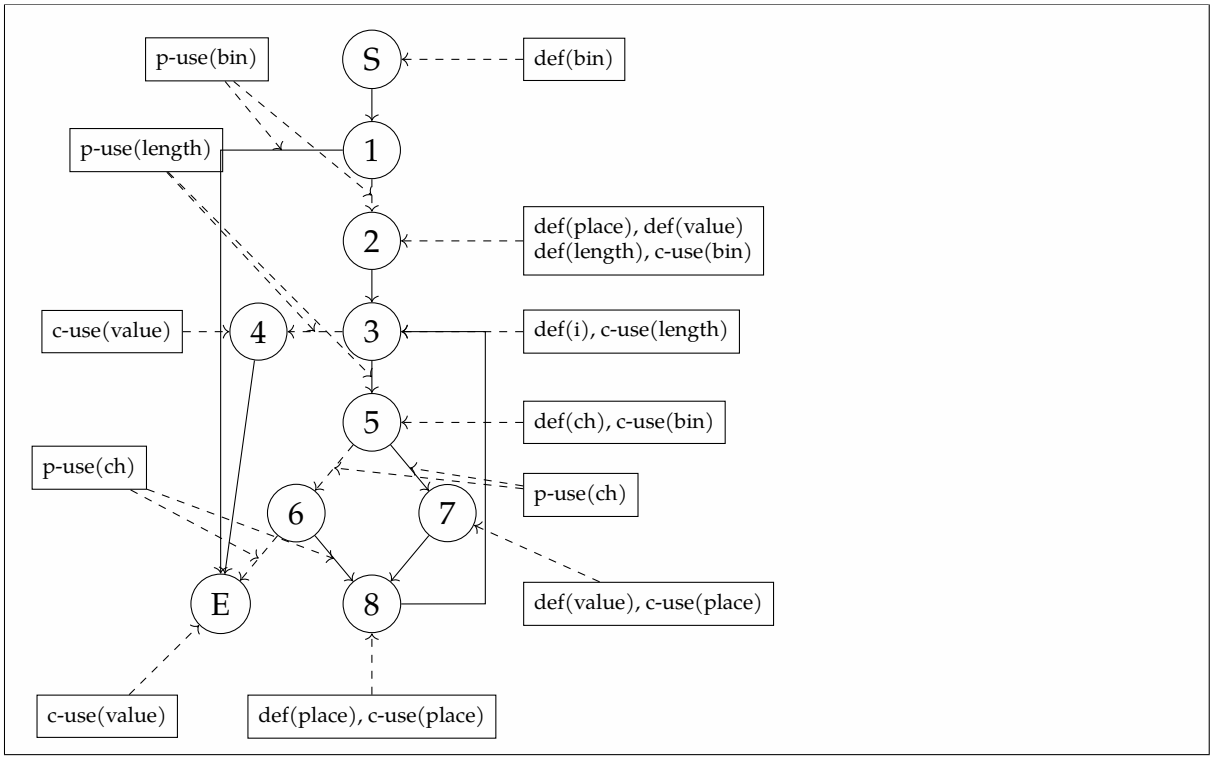
(d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

Lösungsvorschlag

Nein, da **p2** nicht überdeckbar ist.

(e) Geben Sie zu jedem Knoten die jeweilige Datenfluss-Annotation (`defs` bzw. `uses`) für jede betroffene Variable in der zeitlichen Reihenfolge ihres AuftreteS zur Laufzeit an.

Lösungsvorschlag



Aufgabe 4 [wp-Kalkül mit Invariante bei Methode „mul()“]

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei die folgende Methode mit Vorbedingung $P := x \geq 0 \wedge y \geq 0$ und Nachbedingung $Q := x \cdot y = z$.

```
int mul (int x , int y) {
  /* P */
  int z = 0, i = 0;
  while (i++ != x)
    z += y;
  /* Q */
  return z;
}
```

Betrachten Sie dazu die folgenden drei Prädikate:

- $I_1 := z + i \cdot y = x \cdot y$
- $I_2 :=$ false

- $I_3 := z + (x - i) \cdot y = x \cdot y$

(a) Beweisen Sie formal für jedes der drei Prädikate, ob es unmittelbar vor Betreten der Schleife in `mul` gilt oder nicht.

Lösungsvorschlag

$$\begin{aligned} \text{wp}(\texttt{"Code vor der Schleife"}, I_1) &\equiv \text{wp}(\texttt{"int z = 0, i = 0;"}, z + i \cdot y = x \cdot y) \\ &\equiv \text{wp}(\texttt{""}, 0 + 0 \cdot y = x \cdot y) \\ &\equiv 0 = x \cdot y \\ &\equiv \text{falsch} \end{aligned}$$

$$\begin{aligned} \text{wp}(\texttt{"Code vor der Schleife"}, I_2) &\equiv \text{wp}(\texttt{"int z = 0, i = 0;"}, \text{false}) \\ &\equiv \text{wp}(\texttt{""}, \text{false}) \\ &\equiv \text{false} \\ &\equiv \text{falsch} \end{aligned}$$

$$\begin{aligned} \text{wp}(\texttt{"Code vor der Schleife"}, I_3) &\equiv \text{wp}(\texttt{"int z = 0, i = 0;"}, z + (x - i) \cdot y = x \cdot y) \\ &\equiv \text{wp}(\texttt{""}, 0 + (x - 0) \cdot y = x \cdot y) \\ &\equiv x \cdot y = x \cdot y \\ &\equiv \text{wahr} \end{aligned}$$

(b) Weisen Sie formal nach, welche der drei Prädikate Invarianten des Schleifenrumpfs in `mul` sind oder welche nicht.

Lösungsvorschlag

Für den Nachweis muss der Code etwas umformuliert werden:

```
int mul (int x , int y) {
  /* P */
  int z = 0, i = 0;
  while (i != x) {
    i = i + 1;
    z = z + y;
  }
  /* Q */
  return z;
}
```

$$\begin{aligned}
\text{wp}(\texttt{"Code Schleife"}, I_1 \wedge i \neq x) &\equiv \text{wp}(\texttt{"i = i + 1; z = z + y;"}, z + i \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\texttt{"i = i + 1;"}, z + y + i \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\texttt{""}, z + y + (i+1) \cdot y = x \cdot y \wedge i + 1 \neq x) \\
&\equiv z + y + (i+1) \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv z + i \cdot y + 2 \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv \text{falsch} \wedge i + 1 \neq x \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\texttt{"Code Schleife"}, I_2 \wedge i \neq x) &\equiv \text{wp}(\texttt{"i = i + 1; z = z + y;"}, \text{false} \wedge i \neq x) \\
&\equiv \text{wp}(\texttt{""}, \text{false} \wedge i \neq x) \\
&\equiv \text{falsch} \wedge i \neq x \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\texttt{"Code Schleife"}, I_3 \wedge i \neq x) &\equiv \text{wp}(\texttt{"i = i + 1; z = z + y;"}, z + (x - i) \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\texttt{"i = i + 1;"}, z + y + (x - i) \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\texttt{""}, z + y + (x - i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x) \\
&\equiv z + y + x \cdot y - i \cdot y + y = x \cdot y \wedge i + 1 \neq x \\
&\equiv z + 2 \cdot y + x \cdot y - i \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv \text{wahr}
\end{aligned}$$

(c) Beweisen Sie formal, aus welchen der drei Prädikate die Nachbedingung gefolgert werden darf bzw. nicht gefolgert werden kann.

Lösungsvorschlag

$I_1 := z + i \cdot y = x \cdot y$ $I_2 := \text{false}$ $I_3 := z + (x - i) \cdot y = x \cdot y$

$$\begin{aligned}
\text{wp}(\texttt{"Code nach Schleife"}, I_1 \wedge i = x) &\equiv \text{wp}(\texttt{""}, z + i \cdot y = x \cdot y \wedge i = x) \\
&\equiv z + i \cdot y = x \cdot y \wedge i = x \\
&\equiv z + x \cdot y = x \cdot y \\
&\neq Q
\end{aligned}$$

$$\begin{aligned}\text{wp}(\texttt{"Code nach Schleife"}, I_2 \wedge i = x) &\equiv \text{wp}(\texttt{""}, \text{false} \wedge i = x)\\ &\equiv \text{false} \wedge i = x\\ &\equiv \text{falsch}\\ &\neq Q\end{aligned}$$

$$\begin{aligned}\text{wp}(\texttt{"Code nach Schleife"}, I_3 \wedge i = x) &\equiv \text{wp}(\texttt{""}, z + (x - i) \cdot y = x \cdot y \wedge i = x)\\ &\equiv z + (x - i) \cdot y = x \cdot y \wedge i = x\\ &\equiv z + (x - x) \cdot y = x \cdot y\\ &\equiv z + 0 \cdot y = x \cdot y\\ &\equiv z + 0 = x \cdot y\\ &\equiv z = x \cdot y\\ &\equiv Q\end{aligned}$$

(d) Skizzieren Sie den Beweis der totalen Korrektheit der Methode `mul`. Zeigen Sie dazu auch die Terminierung der Methode.

Lösungsvorschlag

Aus den Teilaufgaben folgt der Beweis der partiellen Korrektheit mit Hilfe der Invariante i_3. i steigt streng monoton von 0 an so lange gilt $i \neq x$. $i = x$ ist die Abbruchbedingung für die bedingte Wiederholung. Dann terminiert die Methode. Die Methode `mul` ist also total korrekt.