UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

Software Engineering im Wintersemester 2021/2022
Prof. Dr. Martin Leucker, Malte Schmitz, Stefan Benox, Julian Schulz, Benedikt Stepanek, Friederike Weilbeer, Tom Wetterich

# Übungszettel 4 (Lösungsvorschlag)

22.11.2021

*Abgabe bis Donnerstag, 18. November um 23:59 Uhr online im Moodle.*

## Aufgabe 4.1: Sequenzdiagramm

**4 Punkte, mittel**

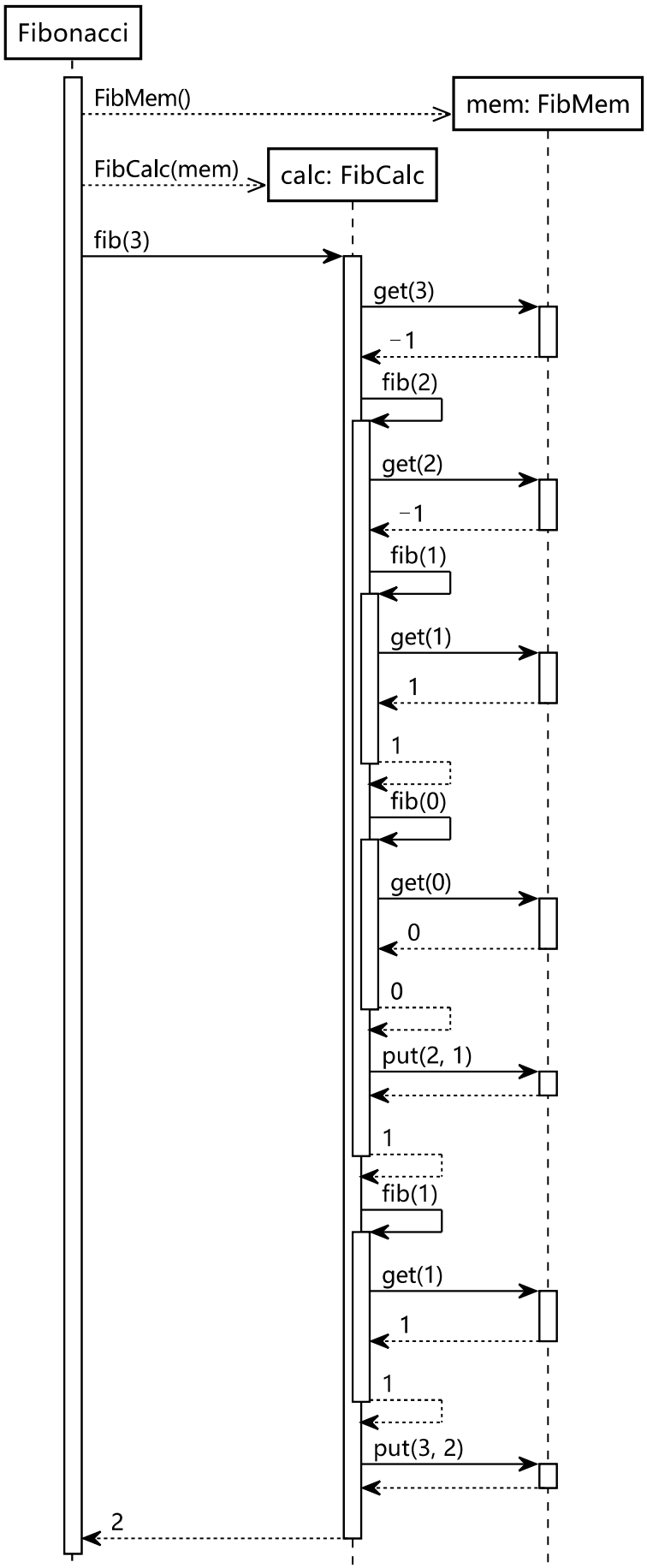
Gegeben ist folgendes Java-Programm zur Berechnung der Fibonacci-Zahlen mit Memoization, d.h. bereits errechnete Teilergebnisse werden zwischengespeichert. Zeichnen Sie ein Squenzdiagramm für den Aufruf der Methode `main` in der Klasse `Fibonacci`, dass folgende Objekte zeigt:

- Die statische Klasse `Fibonacci`,
- die Instanz `calc` von `FibCalc` und
- die Instanz `mem` von `FibMem`.

```
public class Fibonacci {
    public static void main(String[] args) {
        FibMem mem = new FibMem();
        FibCalc calc = new FibCalc(mem);
        calc.fib(3);
    }
}

public class FibCalc {
    private final FibMem mem;

    public FibCalc(FibMem mem) {
        this.mem = mem;
    }

    public int fib(int n) {
```

```
            int v = mem.get(n);
            if (v < 0) {
                v = fib(n - 1) + fib(n - 2);
                mem.put(n, v);
            }
            return v;
        }
    }

    import java.util.HashMap;
    import java.util.Map;

    public class FibMem {
        private Map<Integer, Integer> map =
                new HashMap<>(Map.of(0, 0, 1, 1));
        public int get(int key) {
            return map.getOrDefault(key, -1);
        }
        public void put(int key, int value) {
            map.put(key, value);
        }
    }
```

▼ Lösungsvorschlag




▶ Quelltext des Diagramms

# Aufgabe 4.2: LTL über linearen Läufen

**4 Punkte, mittel**

Geben Sie für die folgenden Eigenschaften LTL-Formeln an, sodass geanu die Läufe, die die jeweilige Eigenschaft erfüllen auch Modell der zugehörigen Formel sind. Hierbei gilt für die Menge der Propositionen $\mathbf{AP} = \{a, b, c\}$ und für das Alphabet $\Sigma = 2^{\mathbf{AP}}$.

1. An der vierten Position gilt $a$.

   ▼ **Lösungsvorschlag**

   Die vierte Position im Wort kann durch dreifache Anwendung des Next-Operators beschrieben werden:

   $$\mathcal{X}\,\mathcal{X}\,\mathcal{X}\,a$$

2. Nach der zweiten Position gilt $a$ nie.

   ▼ **Lösungsvorschlag**

   Die Proposition $a$ gilt nie kann durch $\mathcal{G}\,\neg a$ ausgedrückt werden. Durch vorangestellte Next-Operatoren wird diese Eigenschaft erst ab der dritten Position eingefordert:

   $$\mathcal{X}\,\mathcal{X}\,\mathcal{G}\,\neg a$$

3. Es muss so lange $b$ gelten, bis $a$ und $b$ gleichzeitig gelten.

   ▼ **Lösungsvorschlag**

   Diese Eigenschaft entspricht genau dem Release-Operator:

   $$a\,\mathcal{R}\,b$$

4. Das Wort ist in der Sprache $\Sigma^{\omega}$.

   ▼ **Lösungsvorschlag**

Die Sprache $\Sigma^\omega$ enthält alle unendlichen Worte über dem Alphabet $\Sigma$, sodass die gesuchte LTL-Formel keine Worte ausschließt:

$$\mathbf{true}$$

5. Im ersten Schritt muss $a$ gelten und es muss irgendwann $b$ gelten.

▼ **Lösungsvorschlag**

Aussagenlogische Formeln ohne Temporal-Operatoren beziehen sich jeweils nur auf die erste Position im Wort. Entsprechend prüft die Formel $a$, ob in der ersten Position im Wort $a$ gilt. Der zweite Teil der Eigenschaft entspricht genau dem Finally-Operator. Die Konjunktion beider Eigenschaften kann direkt ausgedrückt werden:

$$a \wedge \mathcal{F}\, b$$

6. Das Wort ist in der Sprache $\emptyset$.

▼ **Lösungsvorschlag**

Die Sprache $\emptyset$ ist leer. Wir suchen also eine kontradiktorische LTL-Formel, also eine, die für kein Wort erfüllt ist:

$$\mathbf{false}$$

7. Immer wenn $a$ gilt muss ab dem nächsten Schritt so lange $b$ gelten bis $c$ gilt.

▼ **Lösungsvorschlag**

Es soll in jedem Schritt überprüft werden, ob $a$ eine weitere Eigenschaft $\varphi$ impliziert. Dies kann durch die Formel $\mathcal{G}(a \rightarrow \varphi)$ erfolgen. Nun soll nicht im gleichen Schritt, sondern im nächsten Schritt erfüllt sein, dass $b$ gilt, bis $c$ gilt:

$$\mathcal{G}(a \rightarrow \mathcal{X}(b\,\mathcal{U}\,c))$$

8. Es darf höchstens zwei mal $a$ gelten.

▼ Lösungsvorschlag

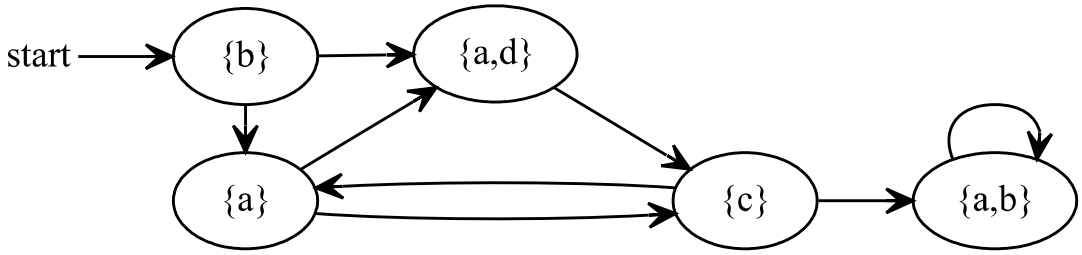
Wir können diese Eigenschaft in LTL beschreiben, indem wir verlangen, dass es nach dem ersten Vorkommen von $a$ höchstens noch ein weiteres $a$ gibt. Nach diesem weiteren $a$ darf es dann kein weiteres $a$ mehr geben.

$$\mathcal{G}(a \rightarrow \mathcal{X}\,\mathcal{G}(a \rightarrow \mathcal{X}\,\mathcal{G}\,\neg a))$$

# Aufgabe 4.3: LTL und Transitionssysteme

**4 Punkte, mittel**

Gegeben sind ein Transitionssystem und folgende LTL-Formeln. Begründen Sie für jede Formel, ob diese auf *allen* möglichen Läufen des Transitionssystems erfüllt ist.




▶ Quelltext des Diagramms

1. $a$

▼ Lösungsvorschlag

Nein. Die Formel beschreibt nur den ersten Zustand und dort gilt die Proposition $a$ nicht, also ist die Formel auf allen Läufen nicht erfüllt.

2. $\mathcal{X}\,d$

▼ Lösungsvorschlag

Nein. Die Formel ist zum Beispiel auf allen Läufen nicht erfüllt, die mit $\{b\}\{a\}$ beginnen.

3. $\mathcal{G}\,\mathcal{F}\,a$

▼ **Lösungsvorschlag**

Ja. Die Formel ist auf allen Läufen erfüllt, denn in jedem möglichen Lauf sind unendlich viele Zustände enthalten, in denen $a$ gilt.

4. $(\neg c)\,\mathcal{U}\,a$

▼ **Lösungsvorschlag**

Ja. Die ersten beiden Zustände sind entweder $\{b\}\{a, d\}$ oder $\{b\}\{a\}$ und in beiden Fällen gilt im zweiten Zustand $a$ und bis dahin gilt $\neg c$, sodass diese Formel auf allen möglichen Läufen erfüllt ist.

5. $\mathcal{F}\,\mathcal{G}(a \vee c)$

▼ **Lösungsvorschlag**

Ja. In allen Zuständen außer dem ersten Zustand gilt $a \vee c$. In jedem Lauf ist der erste Zustand nur genau einmal ganz am Anfang enthalten, sodass nach dem ersten Zustand $a \vee c$ in jedem weiteren Zustand aller möglicher Läufe erfüllt ist. Damit ist die Formel auf allen möglichen Läufen erfüllt.

6. $b \wedge \mathcal{X}(\mathcal{G}(\neg c \rightarrow \neg \mathcal{X} b))$

▼ **Lösungsvorschlag**

Nein. Ein Gegenbeispiel für diese Formel wäre ein Lauf, in dem auf einen Zustand, in dem $\neg c$ gilt, ein Zustand folgt, in dem $b$ gilt. Das ist bei allen Läufen, die in $\{a, b\}^\omega$ enden, unendlich oft der Fall. Also ist diese Formel nicht auf allen Läufen erfüllt.

7. $\mathcal{X}((\mathcal{X}\,b)\,\mathcal{R}\,\mathcal{F}\,c)$

▼ **Lösungsvorschlag**

Ja. In jedem Lauf gilt im dritten oder vierten Zustand $c$. Entsprechend ist $\mathcal{X}\,\mathcal{F}\,c$ in jedem Lauf erfüllt, sodass auch die gegebene Formel in jedem Lauf erfüllt ist.

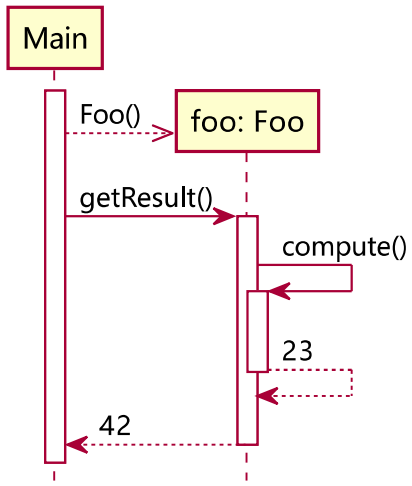
8. $(\mathcal{G}\,\mathbf{true}) \vee \mathcal{X}\,b$

▼ Lösungsvorschlag

Ja. Die Formel $\mathbf{true}$ ist immer erfüllt, also ist auch $\mathcal{G}\,\mathbf{true}$ immer erfüllt. Damit ist auch die gegebene Formel immer erfüllt, auch wenn $\mathcal{X}\,b$ auf keinem möglichen Lauf erfüllt ist.

# Hinweise zum Erstellen der Diagramme

UML-Sequenzdiagramme lassen sich sehr elegant mit PlantUML erstellen, was in CodiMD (oder HedgeDoc) direkt eingebunden ist.




▶ Quelltext des Diagramms