Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. Bernhard Beckert
Thorsten Bormer, Dr. Vladimir Klebanov, Dr. Mattias Ulbrich

Formale Systeme WS 2014/2015

Musterlösung zum Zwischentest 4 am 06.02.2015

Aufgabe 1 (3 Punkte)

Kreuzen Sie an, ob die folgenden Aussagen richtig oder falsch sind. Für jede richtige Antwort erhalten Sie einen Punkt, für jede falsche wird ein Punkt abgezogen. Sie können jedoch nicht weniger als 0 Punkte für diese Aufgabe erhalten. Wenn Sie bei einer Frage kein Kreuz setzen, so wird diese nicht gewertet.

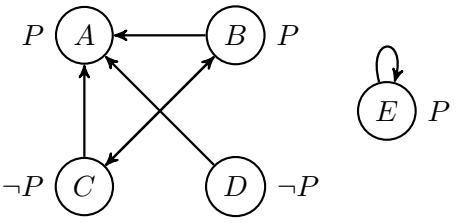
	Richtig	Falsch
1. Jedes Termersetzungssystem, das die Gleichung $x \doteq c$ enthält, ist konfluent (c ist eine Konstante, x eine Variable).	X	
2. Wenn die Spezifikation einer Java-Methode die JML-Klausel `assignable this.x;` enthält, dann darf diese Methode den Wert des Feldes `x` für das Objekt `this` verändern.	X	
3. Die modallogische Formel $\Box P \to \Diamond P$ ist eine Tautologie in allen transitiven Kripkerahmen.		X

Zur Begründung:

1. Mit der Gleichung kann jeder Term in einem Schritt zu c reduziert werden.
2. Das ist gerade die Bedeutung dieser Klausel.
3. Der (transitive) Kripkerahmen mit nur einer Welt, von der aus nichts zu erreichen ist, ist ein Gegenbeispiel.

Aufgabe 2 (3 Punkte)

Gegeben sei die folgende Kripke-Struktur über der Signatur $\Sigma = \{P\}$:



Geben Sie für die folgenden Formeln jeweils genau diejenigen der Welten A, B, C, D, E an, in denen die Formel wahr ist.

(a) $\Box P \to P$ A, B, E

(b) $\Box\Diamond P$ A, E

(c) $\Box(\neg P \wedge P)$ A

Aufgabe 3 (4 Punkte)

Gegeben sei die folgende Java-Klasse, die mit einer noch unvollständigen JML-Spezifikation versehen ist.

```
public class C {
    int i;

    /*@ public normal_behavior
      @
      @ requires i>0;
      @
      @ ensures
      @
      @
      @
      @
      @
      @*/
    public int m(int k) {

        int j = 0;

        /*@ loop_invariant
          @
          @
          @
          @
          @
          @
          @*/
        while (i>0) {
            i--;
            j = j+k;
        }

        return j;
    }
}
```

a. Geben Sie eine Nachbedingung (`ensures`) für die Methode `m()` an, so dass die Implementierung von `m()` die JML-Spezifikation erfüllt.
Ihre Nachbedingung soll so stark sein, dass sie den Rückgabewert der Methode eindeutig bestimmt.

```
ensures \result == \old(i) * k;
```

b. Geben Sie eine Schleifeninvariante (`loop_invariant`) für die Schleife an.
Ihre Schleifeninvariante soll so stark sein, dass aus der Invarianten und der Terminierung der Schleife die Nachbedingung folgt.

```
loop_invariant (i >= 0) && j == (\old(i)-i) * k;
```