Klausur Formale Systeme

Fakultät für Informatik

SS 2018

Prof. Dr. Bernhard Beckert

30. Juli 2018

Name: _____

Vorname: _____

Matrikel-Nr.: _____

Die Bearbeitungszeit beträgt 60 Minuten.

A1 (13)	A2 (8)	A3 (6)	A4 (8)	A5 (9)	A6 (9)	A7 (7)	Σ (60)

Bewertungstabelle bitte frei lassen!

Gesamtpunkte:

1 Zur Einstimmung (5+5+3 = 13 Punkte)

a. Kreuzen Sie in der folgenden Tabelle alles Zutreffende an.

Für jede korrekte Antwort gibt es einen Punkt, **für jede falsche Antwort wird ein *halber Punkt* abgezogen!** (Dabei werden jedoch keinesfalls weniger als 0 Punkte für diese Teilaufgabe vergeben.)

Hinweise:

- „PL1“ steht für „Prädikatenlogik erster Stufe (mit Gleichheit $\doteq$)“, wie sie in der Vorlesung vorgestellt wurde. Auf diese beziehen sich in Teilaufgabe a. auch die Begriffe „erfüllbar“, „allgemeingültig“ und „unerfüllbar“.
- p, q sind Prädikatensymbole, f, g sind Funktionssymbole und x, y, z sind Variablen.
- Es gelten die üblichen Klammereinsparungsregeln.

	keine Formel der PL1	allgemeingültig	erfüllbar, aber nicht allgemeingültig	unerfüllbar
$\exists x\, \exists y\, \neg x \doteq y$			X	
$(\forall x\, q(x)) \doteq (\forall x\, q(x))$	X			
$\forall x \forall y \big((p(x,y) \wedge p(y,x)) \rightarrow x \doteq y\big)$			X	
$(\forall x\, x \doteq f(g(x))) \rightarrow \big((\forall y\, p(f(y))) \rightarrow (\forall z\, p(z))\big)$		X		
$\exists x \forall y\, \neg(f(x) \doteq y)$				X

b. Bitte kreuzen Sie in der folgenden Tabelle das Zutreffende an. Für korrekte Antworten erhalten Sie einen Punkt, **für falsche Antworten wird *ein Punkt* abgezogen**. Dabei werden jedoch nie weniger als 0 Punkte für diese Teilaufgabe vergeben.

	Richtig	Falsch
Seien r, tc_r zweistellige Prädikatensymbole. Es gibt eine prädikatenlogische (PL1) Formel, die folgendes formalisiert: *tc_r ist die transitive Hülle von r.*		X
Eine JML-assignable-Klausel ist erfüllt, wenn höchstens die Speicherstellen angegeben sind, welche von der nachfolgenden Methode verändert werden.		X
Aus der Vollständigkeit des prädikatenlogischen Tableaukalküls folgt, dass das prädikatenlogische Erfüllbarkeitsproblem entscheidbar ist.		X
Eine korrekte JML-Schleifeninvariante muss vor Beginn der ersten Schleifeniteration sowie nach jeder weiteren Schleifeniteration gelten.	X	
Es ist entscheidbar, ob eine beliebige LTL-Formel allgemeingültig ist.	X	

Fortsetzung 1 Zur Einstimmung

c. Sei **XU** ein LTL-Operator mit folgender Definition (ϕ, ψ sind beliebige LTL-Formeln):

$$\phi \,\mathbf{XU}\, \psi \quad =_{\text{def}} \quad \mathbf{X}(\phi \,\mathbf{U}\, \psi) \ .$$

Zeigen Sie, dass die LTL-Operatoren **X** und **U** sich mit den Operatoren aus der Menge

$$\{\mathbf{XU}, \wedge, \vee, \neg, \mathbf{0}, \mathbf{1}\}$$

darstellen lassen!

$\mathbf{X}\alpha \equiv \mathbf{0}\,\mathbf{XU}\,\alpha$	1 P.
$\alpha \,\mathbf{U}\, \beta \equiv (\alpha \wedge \alpha \,\mathbf{XU}\, \beta) \vee \beta$	2 P.

2 Formale Methoden in der Praxis (4+4 = 8 Punkte)

a. Sie arbeiten für einen Automobilhersteller, bei dem die Kunden ein Automobil vor dem Kauf durch auswählen von Optionen konfigurieren können. Leider sind nicht alle Konfigurationen zulässig. Sie müssen die möglichen Konfigurationen sowie die Zulässigkeitsbedingungen in einer Logik formalisieren und mit einem Verfahren automatisch überprüfen, ob eine Konfiguration zulässig ist. Welche Logik und welches Verfahren wählen Sie? Begründen Sie Ihre Logik- und Verfahrensauswahl!

Logik: Aussagenlogik

Begründung: Da endlich viele Optionen, reicht Aussagenlogik völlig aus.

Verfahren: DPLL

Begründung: Automatisch, entscheidbar, funktioniert gut in der Praxis.

b. Der Automobilhersteller verfügt über Produktionsanlagen, bei denen Menschen und Maschinen zusammenarbeiten. Damit die Menschen in Sicherheit arbeiten, dürfen die Operationen der Maschinen nur in genau festgelegten Reihenfolgen durchgeführt werden. Sie werden beauftragt, die Sicherheitsanforderungen in einer Logik zu formalisieren, die möglichen Operationen zu modellieren und mit einem Verfahren automatisch zu überprüfen, ob die Sicherheitsanforderungen erfüllt sind. Welche Logik und welches Verfahren wählen Sie? Begründen Sie Ihre Logik- und Verfahrensauswahl!

Logik: LTL

Begründung: Reihenfolgen der Operationen sind temporale Eigenschaften, die man mit LTL formalisieren kann.

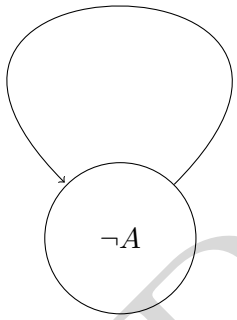
Verfahren: Modellüberprüfung

Begründung: Automatisch, entscheidbar.

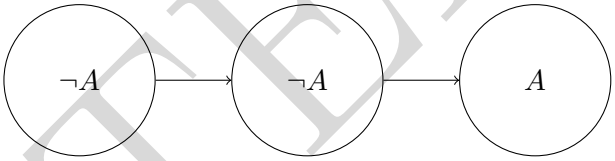
3 Modallogik (2+2+2 = 6 Punkte)

Zeichnen Sie zu jeder der drei folgenden modallogischen Formeln F_i jeweils eine Kripkestruktur K_i, die **kein Modell** von F_i ist, für die also **nicht** $K_i \models F_i$ gilt. Dabei sind A und B aussagenlogischen Variablen. Geben Sie zu jeder Welt von K_i explizit an, ob die aussagenlogischen Variablen dort jeweils wahr oder falsch sind.

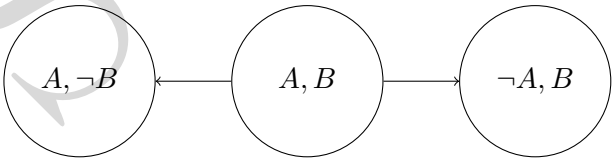
a. F_1: $(\Box A) \wedge (\neg \Diamond A)$



b. F_2: $(\Box\Box A) \rightarrow \Box A$



c. F_3: $((\Diamond A) \wedge (\Diamond B)) \rightarrow \Diamond(A \wedge B)$



4 Formalisieren in PL1 (2+2+2+2 = 8 Punkte)

Gegeben sei die prädikatenlogische Signatur

$$\Sigma = (\{inst, sub, cls, obj\}, \{Object\}, \alpha).$$

Sie enthält die einstelligen Prädikatensymbole $cls(\cdot)$ und $obj(\cdot)$, die zweistelligen Prädikatensymbole $inst(\cdot,\cdot)$, $sub(\cdot,\cdot)$, sowie die Konstante *Object*.

Zur Auswertung der Formeln werden nur solche Interpretationen (D, I) über Σ verwendet, in denen

- das Universum D eine Menge von Java-Objekten (nicht `null`) und Klassen ist,
- das Prädikat $obj(x)$ genau dann wahr ist, wenn x ein Objekt ist,
- das Prädikat $cls(x)$ genau dann wahr ist, wenn x eine Klasse ist,
- das Prädikat $inst(x, y)$ genau dann wahr ist, wenn x eine Instanz von y ist,
- das Prädikat $sub(x, y)$ genau dann wahr ist, wenn x ein Untertyp von y ist.

Geben Sie jeweils eine Formel der Prädikatenlogik mit Gleichheit über Σ an, die folgende Sachverhalte darstellt:

a. Jede Klasse ist ein Untertyp von *Object*.

$\boxed{(\forall x\ cls(x) \rightarrow sub(x, Object))}$

---

b. Ist o ein beliebiges Objekt und c eine beliebige Klasse, so dass o eine Instanz von c ist, dann ist o auch Instanz aller Klassen, von denen c ein Untertyp ist.

$\boxed{\forall o\, \forall c\, \big((obj(o) \land cls(c) \land inst(o, c)) \rightarrow (\forall t\, (cls(t) \land sub(c, t)) \rightarrow inst(o, t))\big)}$

---

c. Ist ein beliebiges Objekt o Instanz einer Klasse c, dann ist o verschieden von allen Objekten, die nicht Instanz von c sind.

$\boxed{\forall o\, \forall c\big((obj(o) \land cls(c)) \rightarrow \forall o'\, ((obj(o') \land \neg inst(o', c)) \rightarrow \neg o \doteq o')\big)}$

---

d. Es gibt eine Klasse, die genau ein Instanz-Objekt hat.

$\boxed{\exists c\, \big(cls(c) \land \exists o\, (obj(o) \land inst(o, c) \land \forall p\, ((obj(p) \land inst(p, c)) \rightarrow o \doteq p))\big)}$

---

5 Resolutionskalkül (9 Punkte)

Zeigen Sie mit Hilfe des Resolutionskalküls für die Prädikatenlogik, dass folgende Klauselmenge unerfüllbar ist. Notieren Sie bei jedem Schritt die Klauseln auf denen die Resolutionsregel angewandt wurde, sowie **die verwendete Substitution**.

Signatur: p, q, r sind Prädikate; b ist eine Konstante; f ist eine Funktion; $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ sind Variablen.

Hinweis: Mit der allgemeinen Resolutionsregel können die Klauseln (1) und (2) so resolviert werden, dass eine Einerklausel entsteht.

(1) $\{p(x_1, f(x_2)),\ p(f(x_2), x_1)\}$ (2) $\{\neg p(x_3, f(f(x_4))),\ q(x_4, x_3)\}$

(3) $\{\neg q(x_5, f(x_6)),\ r(x_6)\}$ (4) $\{\neg q(b, x_7),\ \neg r(f(x_7))\}$

(5) $\{q(x_4, f(f(x_4)))\}$ aus $1, 2$ mit $\sigma = \{x_1 \setminus f(f(x_4)),\ x_2 \setminus f(x_4),\ x_3 \setminus f(f(x_4))\}$

(6) $\{r(f(x_4))\}$ aus $5, 3$ mit $\sigma = \{x_5 \setminus x_4,\ x_6 \setminus f(x_4)\}$

(7) $\{\neg q(b, x_7)\}$ aus $6, 4$ mit $\sigma = \{x_4 \setminus x_7\}$

(8) $\square$ aus $5, 7$ mit $\sigma = \{x_4 \setminus b, x_7 \setminus f(f(b))\}$

6 Spezifikation mit der Java Modeling Language

(2+4+3 = 9 Punkte)

a. Geben Sie die Bedeutung der folgenden JML-Klasseninvariante in natürlicher Sprache wieder.

```
public class A {
  public int[] a;

  /*@ public invariant
    @   a != null &&
    @   (\forall int i; 0 <= i && i < a.length;
    @     (\forall int j, k; 0 <= j && 0 <= k && j < k && k < i; a[j] * a[k] != a[i]));
    @*/
}
```

Das Array a ist von `null` verschieden und kein Array-Element lässt sich durch das Multiplizieren von zwei vorhergehenden Array-Elementen, die an jeweils voneinander verschiedenen Positionen stehen, berechnen.

b. Sei weiterhin eine Methode `m` in der obigen Klasse `A`. Vervollständigen Sie den nachstehenden JML-Methodenvertrag für `m`, so dass er Folgendes besagt:

Wenn das Array `a` nur positive Werte enthält, terminiert die Methode und wirft keine Exception, ändert keine Speicherstellen, und erfüllt folgende Nachbedingungen:

- Die Methode liefert genau dann einen positiven Wert zurück, wenn `a` mindestens drei direkt nebeneinanderliegende Elemente mit dem gleichen Wert beinhaltet. Der Rückgabewert ist dann der Wert eines dieser Elemente (da alle drei Elemente den gleichen Wert haben, können Sie hier ein beliebiges dieser drei Elemente wählen).
- Genau dann wenn keine solche wertgleiche Dreiergruppe im Array `a` existiert, wird der Wert 0 zurückgegeben.

```
/*@
  @
  @
  @
  @
  @
  @
  @
  @
  @
  @
  @*/
public int m() { ... }
```

Fortsetzung 6 Spezifikation mit der Java Modeling Language

```
/*@ public normal_behaviour
  @ requires (\forall int i; 0 <= i && i < a.length; 0 < a[i]);
  @ ensures \result == 0 <==>
  @  (\forall int i; 0 < i && i < a.length - 1;
  @    a[i - 1] != a[i] || a[i] != a[i + 1]);
  @ ensures 0 < \result <==>
  @  (\exists int i; 0 <= i && i < a.length - 2;
  @     a[i] == \result && a[i + 1] == \result
  @  && a[i + 2] == \result);
  @ assignable \nothing;
  @*/
```

c. Sei weiterhin eine Methode `arrayMax` der obigen Klasse `A` durch die untenstehende Implementierung gegeben. Der untenstehende Vertrag spezifiziert, dass die Methode `arrayMax` das maximale Element von `a`, wenn `a` nicht leer ist, zurückgibt. Die untenstehende Schleifeninvariante ist unvollständig. Ergänzen Sie diese, sodass daraus eine korrekte Invariante entsteht, mit der die Nachbedingung des Methodenvertrags bewiesen werden kann.

```
/*@ normal_behaviour
  @   requires (\forall int i; 0 <= i && i < a.length; 0 <= a[i]);
  @   ensures (\forall int i; 0 <= i && i < a.length; a[i] <= \result);
  @   ensures (0 < a.length
  @          ==> (\exists int i; 0 <= i && i < a.length; \result == a[i]));
  @   assignable \nothing;
  @*/
int arrayMax() {
    int max = 0;
    /*@ loop_invariant 0 <= k && k <= a.length && (k == 0 ==> max == 0);
      @
      @
      @
      @
      @
      @
      @
      @
      @
      @  assignable \nothing;
      @  decreases a.length - k;
      @*/
    for(int k = 0; k < a.length; k++) { if(max < a[k]) max = a[k]; }
    return max;
}
```

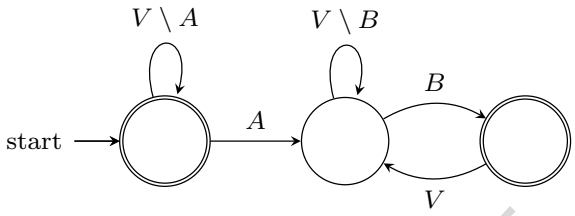
```
/*@ loop_invariant (\forall int i; 0 <= i && i < k; a[i] <= max)
  @   && (0 < k ==> (\exists int i; 0 <= i && i < k; max == a[i]));
  @*/
```

7 Lineare Temporale Logik (LTL) und Büchiautomaten

(2+2+3 = 7 Punkte)

Gegeben sei folgender Büchi-Automat $\mathcal{A}$ über dem Alphabet V und der AL-Signatur $\Sigma = \{a, b\}$. Wir verwenden die aus der Vorlesung bekannten Abkürzungen:

$$V = \{\, \{\,\},\ \{a\},\ \{b\},\ \{a, b\}\,\} \qquad A = \{M \in V \mid a \in M\} \qquad B = \{M \in V \mid b \in M\}$$



a. Geben Sie an, ob $L_i \subseteq L^\omega(\mathcal{A})$ gilt:

$L_1 = B^*A^\omega$ ☐ Ja ☒ Nein

$L_2 = A^*(BA^*)^\omega$ ☒ Ja ☐ Nein

b. Beschreiben Sie in natürlicher Sprache die Omega-Strukturen in $L^\omega(\mathcal{A})$:

$\mathcal{A}$ akzeptiert alle Wörter, in denen A niemals gilt oder unendlich oft B gilt.

c. Geben Sie eine LTL-Formel ϕ über der Signatur $\{a, b\}$ an, die zum obigen Büchi-Automaten $\mathcal{A}$ äquivalent ($\xi \models \phi \Leftrightarrow \xi \in L^\omega(\mathcal{A})$) ist.

$\phi =$ $\Diamond a \rightarrow \Box\Diamond b \quad \equiv \quad \Box\neg a \vee \Box\Diamond b$