Klausur Formale Systeme

Fakultät für Informatik

WS 2017/2018

Prof. Dr. Bernhard Beckert

28. Februar 2018

Vorname: **Vorname**
Name: **Familienname**
Matrikel-Nr.: **Matr.-Nr.**

Hörsaal: **Hörsaal** **Sitzplatz**
Code: **Nonce**

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:

*Name: **Vorname** **Familienname***
*Matr.-Nr.: **Matr.-Nr.***

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 sind Variablen.
- Es gelten die üblichen Klammereinsparungsregeln.

	keine Formel der PL1	allgemeingültig	erfüllbar, aber nicht allgemeingültig	unerfüllbar
$\forall x\ (p(x) \wedge q(x)) \rightarrow p(q(x))$	X			
$(\forall x\ p(x)) \rightarrow (\forall x\ p(f(x)))$		X		
$\neg\forall x\ \exists y\ y \doteq f(x)$				X
$(\forall x\ p(f(x))) \rightarrow (\forall x\ p(x))$			X	
$\forall x\ (f(g(x)) \doteq x) \rightarrow \forall y\ \forall z\ (g(y) \doteq g(z) \rightarrow y \doteq z)$		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
Sei F_{sko} eine Formel in Skolemnormalform, die aus der prädikatenlogischen Formel F durch Skolemisierung entstanden ist. Dann gilt: $F_{\text{sko}} \models F$	X	
Das ω-Wort $(aba)^\omega$ ist Element des Limes $\lim(K)$, wobei K die Sprache ist, die durch den regulären Ausdruck $(bba)^*bb$ beschrieben wird.		X
Wenn der Wert einer JML-Schleifenvariante nach jeder Schleifeniteration nicht-negativ ist, dann ist die Terminierung der Schleife garantiert.		X
Sei (Σ, E) ein *kanonisches* Termersetzungssystem. Dann ist für beliebige Grundterme s, t entscheidbar, ob $E \models s \doteq t$.	X	
Für jede LTL-Formel ϕ gibt es einen entsprechenden *deterministischen* Büchiautomaten, der genau die Modelle von ϕ akzeptiert.		X

Fortsetzung 1 Zur Einstimmung

c. Gegeben sei die prädikatenlogische Signatur

$$\Sigma = \langle \{c, d\},\ \{p, q\},\ \alpha \rangle \quad \text{mit} \quad \alpha(c) = \alpha(d) = 0 \quad \text{und} \quad \alpha(p) = 2,\ \alpha(q) = 1$$

Das heißt, diese Signatur enthält zwei Konstanten c, d, ein zweistelliges Prädikatensymbol p und ein einstelliges Prädikatensymbol q.

Wie viele verschiedene Herbrand-Interpretationen über der Signatur Σ gibt es? Begründen Sie Ihre Antwort.

Lösung: Die Domäne besteht aus zwei Grundtermen c und d. Es gibt 6 atomare variablenfreie Formeln:

$$p(c, c), p(c, d), p(d, c), p(d, d), q(c), q(d) \ .$$

Die Herbrand-Interpretationen unterscheiden sich (nur) im Wahrheitswert dieser atomaren Formeln, die jeweils wahr oder falsch sein können.

Darum gibt es

$$2^6 = 64$$

verschiedene Herbrand-Interpretationen über Σ.

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

Sie arbeiten in einer Softwarefirma und Ihr Chef möchte Formale Methoden einsetzen, um die Zuverlässigkeit der entwickelten Software zu erhöhen. Sie erinnern sich daran, was Sie in der Vorlesung Formale Systeme gelernt haben, und denken über die Vor- und Nachteile der beiden folgenden Alternativen nach:

1. Spezifikation der Eigenschaften in LTL, Modellierung der Software als Büchi-Automat und Überprüfung mit Model Checking;
2. Spezifikation in JML, Implementierung der Software in Java und Überprüfung mit einem deduktiven Programmverifikationssystem (wie z.B. dem KeY-System).

Nennen Sie zu beiden Alternativen jeweils zwei positive Argumente, also Vorteile gegenüber der jeweils anderen Alternative.

Vorteile LTL + Model Checking

- Entscheidbar
- Gegenbeispiel wird geliefert wenn die Überprüfung fehlschlägt
- Vollautomatisches Verfahren
- LTL hat einen kompakteren Vokabulars als JML
- LTL ist unabhängig von der Programmiersprache
- Bereits in der Modellierungshase anwendbar
- Besser für temporale Eigenschaften

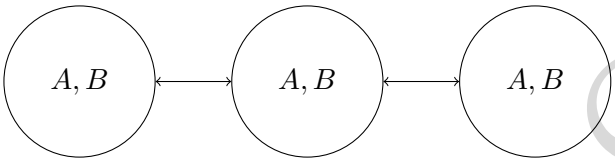
Vorteile JML + KeY:

- Modularität
- Höhere Ausdrucksmächtigkeit
- Kein Modell der Software nötig
- Java-ähnliche Syntax von JML, JML ist einfacher für Programmierer zu lesen, schreiben oder vestehen
- Einfacher zu identifizieren, welcher Software-Teil die Spezifikation verletzt
- besser parallelisierbar

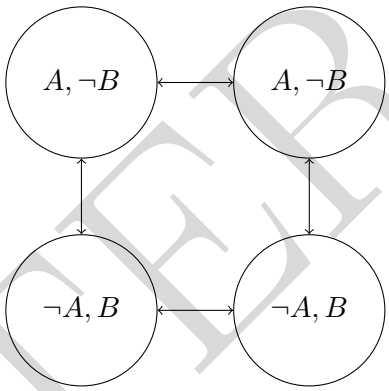
3 Modallogik (2+2+2 = 6 Punkte)

Zeichnen Sie zu jeder der folgenden modallogischen Formeln jeweils eine Kripkestruktur mit **mindestens drei Welten**, so dass die entsprechende Formel in **jeder** Welt wahr ist. Dabei sind A und B aussagenlogische Variablen. Geben Sie zu jeder Welt explizit an, ob die Variablen A, B dort wahr oder falsch sind.

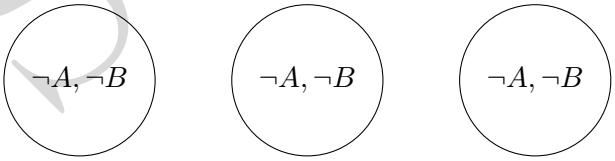
a. $(\Diamond\Box A) \wedge (\Box\Diamond B)$



b. $(\Diamond\neg A) \wedge (\Diamond\neg B) \wedge (\Box(A \vee B))$



c. $(A \leftrightarrow \Diamond B) \wedge (B \leftrightarrow \Diamond\neg A)$



*Name: **Vorname** **Familienname***
*Matr.-Nr.: **Matr.-Nr.***

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

Gegeben sei die prädikatenlogische Signatur $\Sigma = (\{huhn, ei, legt, brütet\}, \{\}, \alpha)$. Sie enthält die einstelligen Prädikatensymbole $huhn(\cdot)$ und $ei(\cdot)$ und die zweistelligen Prädikatensymbole $legt(\cdot,\cdot)$ und $brütet(\cdot,\cdot)$.

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

- das Universum D eine Menge von Hühnern und Eiern ist,
- das Prädikat $huhn(x)$ genau dann wahr ist, wenn x ein Huhn ist,
- das Prädikat $ei(x)$ genau dann wahr ist, wenn x ein Ei ist,
- das Prädikat $legt(x, y)$ genau dann wahr ist, wenn y von x gelegt wurde,
- das Prädikat $brütet(x, y)$ genau dann wahr ist, wenn y von x gebrütet wird.

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

a. Hühner sind keine Eier.

$\boxed{\forall x\,(huhn(x) \rightarrow \neg ei(x))}$

---

b. Es gibt ein Ei, das von keinem Huhn gebrütet wird.

$\boxed{\exists e\,\big(ei(e) \wedge (\forall h\; huhn(h) \rightarrow \neg brütet(h, e))\big)}$

---

c. Hühner brüten nur selbst gelegte Eier.

$\boxed{\forall h\,\big(huhn(h) \rightarrow \forall e\,(ei(e) \rightarrow (brütet(h, e) \rightarrow legt(h, e)))\big)}$

---

d. Zwei verschiedene Hühner brüten nie dasselbe Ei.

$\boxed{\forall h_1\,\big(huhn(h_1) \rightarrow \forall h_2(\; huhn(h_2) \rightarrow \forall e\,(ei(e) \rightarrow ((brütet(h_1, e) \wedge brütet(h_2, e)) \rightarrow h_1 \doteq h_2))))\big)}$

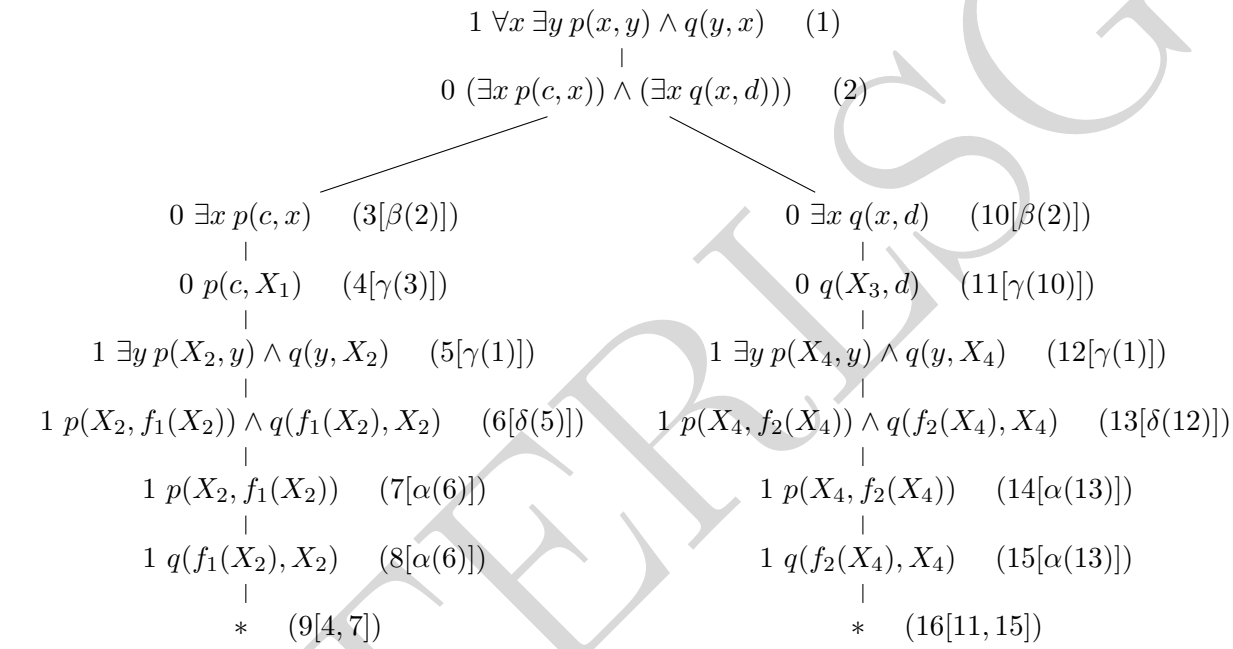
Die Prädikate wie $huhn(h)$ oder $ei(e)$ können nicht ersatzlos wegfallen: Aus $legt(h, e)$ kann nicht abgeleitet werden, dass $ei(e)$ gilt. Für zwei Eier e_1, e_2 ist der Wert von $legt(e_1, e_2)$ nicht definiert.

5 Tableaukalkül (9 Punkte)

a. Vervollständigen und schließen Sie den folgenden Tableau-Beweis.

Notieren Sie dabei:

- den Regeltyp $(\alpha, \beta, \gamma, \delta)$ und die Formel, auf die eine Regel angewendet wird,
- bei Abschlüssen die beiden Partner,
- sowie die schließende Substitution.



Schließende Substitution: $\sigma = \{X_2/c,\ X_1/f_1(c),\ X_4/d,\ X_3/f_2(d)\}$

6 Spezifikation mit der Java Modeling Language

(4+5 = 9 Punkte)

a. Geben Sie in natürlicher Sprache wieder, was der folgende JML-Methodenvertrag für die Methode `m` aussagt.

Hinweis: Der verallgemeinerte JML-Quantor `(\num_of T x; B; R)` zählt die Anzahl der Werte für `x` mit Typ `T` unter Bedingung `B`, die das Prädikat `R` erfüllen. Er ist äquivalent zum Ausdruck `(\sum T x; B && R; 1L)` (das Ergebnis ist vom Typ `long`).
Die Java-Konstanten `Integer.MIN_VALUE` und `Integer.MAX_VALUE` bezeichnen den kleinsten bzw. größten in Java darstellbaren Wert vom Typ `int`.

```
public class A {
  /*@ public normal_behaviour
    @ requires a != null;
    @ assignable \nothing;
    @ ensures
    @    \result.length
    @    == a.length - (\num_of int i; 0 <= i && i < a.length; a[i] == k)
    @ && (\forall int i; 0 <= i && i < \result.length; \result[i] != k)
    @ && (\forall int i; Integer.MIN_VALUE <= i
    @                    && i <= Integer.MAX_VALUE && i != k;
    @      (\num_of int j; 0 <= j && j < a.length; a[j] == i)
    @      == (\num_of int j; 0 <= j && j < \result.length; \result[j] == i));
    @*/
  public int[] m(int[] a, int k) { ... }
}
```

Die Methode `m` gibt für ein Ganzzahl-Array `a` und eine ganze Zahl `k` ein Ganzzahl-Array zurück. Wenn `m` für ein von `null` verschiedenes Array `a` aufgerufen wird, gilt nach Ausführung der Methode:

- Die Länge des zurückgegebenen Arrays ist die Länge von `a` abzüglich der Anzahl der darin enthaltenen Einträge mit Wert `k`.
- Das zurückgegebene Array enthält nur Einträge mit Wert ungleich `k`.
- Die Anzahl der Vorkommen in `a` ist für jeden Java-Ganzzahlwert (d.h. mindestens `Integer.MIN_VALUE` und höchstens `Integer.MAX_VALUE`) ungleich `k` identisch zur Anzahl der Vorkommen des Wertes im zurückgegebenen Array.
- Keine vor Ausführung existierenden Speicherstellen wurden verändert.
- Es ist keine Ausnahme aufgetreten.

Alternativ: Die Methode `m` gibt für ein Ganzzahl-Array `a` und eine ganze Zahl `k` ein Ganzzahl-Array zurück. Falls `a` vor Ausführung nicht `null` ist, terminiert `m` normal, verursacht keine Ausnahmen und ändert keine zuvor existierenden Stellen im Heap. Außerdem gibt `m` ein von `null` verschiedenes Array zurück, in dem `k` nicht vorkommt und alle von `k` verschiedenen Integers genau so häufig vorkommen wie in `a`.

Fortsetzung 6 Spezifikation mit der Java Modeling Language

b. Gegeben sei eine Methode `f`, die eine durch ein Array `a` dargestellte Permutation invertiert. Genauer: Nach Ausführung steht an jeder Position `i` in `a` nun als Wert der Index, an dem im alten Array der Eintrag `i` stand.

Beispiel: Wenn `a = {1, 2, 0}` vor Ausführung galt, gilt nach Ausführung `a = {2, 0, 1}`.

Vervollständigen Sie den nachstehenden JML-Methodenvertrag, so dass er Folgendes besagt:

> Wird die Methode `f` für ein von `null` verschiedenes Array `a` aufgerufen, dann gilt nach Ausführung:
> Das Array `a` enthält an jeder Index-Position `i` als Eintrag genau die Index-Position des Eintrags vom Wert `i` vor Ausführung der Methode.
>
> **Hinweis:** In der Nachbedingung brauchen Sie die Länge des Arrays nicht zu spezifizieren. Gehen Sie davon aus, dass diese unverändert bleibt. Außerdem können Sie davon ausgehen, dass das Array nicht leer ist und dass jeder Wert zwischen 0 (inklusive) und `a.length` (exklusive) als Eintrag vorkommt (bereits unten als Vorbedingung spezifiziert).

```
public class B {
  /*@ spec_public @*/ int[] a;
  /*@ public normal_behaviour
    @  requires 0 < a.length;
    @  requires (\forall int i; 0 <= i && i < a.length;
    @                (\exists int j; 0 <= j && j < a.length; a[j] == i));
    @  ensures
    @
    @
    @
    @
    @
    @
    @
    @
    @
    @
    @*/
    public void f() { ... }
  }
```

Es existieren mehrere korrekte Lösungen, mindestens folgende vier:

i.
```
@  ensures (\forall int i; 0 <= i && i < a.length; a[i] == \old(a[a[i]]));
```

ii.
```
@  ensures (\forall int i; 0 <= i && i < a.length; a[\old(a[i])] == i);
```

iii.
```
@  ensures (\forall int i; 0 <= i && i < a.length;
@            (\forall int j; 0 <= j && j < a.length;
@                      \old(a[i]) == j ==> a[j] == i));
```

iv.
```
@  ensures (\forall int i; 0 <= i && i < a.length;
@            (\exists int j; 0 <= j && j < a.length;
@                      a[i] == j && \old(a[j]) == i));
```

7 Lineare Temporale Logik (LTL) und Büchiautomaten (3+(2+2) = 7 Punkte)

a. Gegeben sei die Signatur $\Sigma = \{p, q, r\}$ und das zugehörige Alphabet $V = 2^{\Sigma}$.

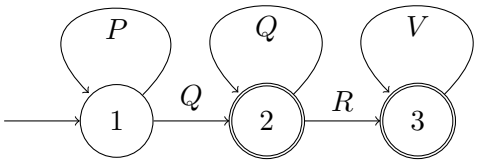
Geben Sie einen Büchi-Automaten $\mathcal{A}$ über dem Alphabet V an, der genau die LTL-Formel

$$p \,\mathbf{U}\, (q \,\mathbf{U_W}\, r)$$

akzeptiert, d.h., so dass $L^{\omega}(\mathcal{A}) = \{\xi \in V^{\omega} : \xi \models p \,\mathbf{U}\, (q \,\mathbf{U_W}\, r)\}$ gilt.

Hinweis: Verwenden Sie dabei folgende Symbolmengen:

$P := \{b \in V \mid p \in b\}$ $\quad$ $Q := \{b \in V \mid q \in b\}$ $\quad$ $R := \{b \in V \mid r \in b\}$



b. Formalisieren Sie folgendene Sachverhalte aus dem Straßenverkehr in LTL.

Dabei sind A, B, C bzw. G, H aussagenlogische Variablen.

i. Spurwechsel:

Wenn Sie die Spur wechseln wollen (A), müssen Sie dies durch Blinken (B) anzeigen. Das Blinken wird aufrechterhalten, bis der Spurwechsel abgeschlossen (C) ist.

$\Box(A \to (B \,\mathbf{U}\, C))$	$(A \to (B \,\mathbf{U}\, C))$	$\Box(A \to B \wedge (B \,\mathbf{U}\, C))$	(2 P)
$(A \to B \wedge (B \,\mathbf{U}\, C))$	$\Box(A \to \Diamond(B \wedge B \,\mathbf{U}\, C))$	$\Box(A \to \mathbf{X}(B \wedge B \,\mathbf{U}\, C))$	(2 P)
$\Box(\mathbf{X}A \to (B \wedge B \,\mathbf{U}\, C))$			(2 P)
$\Box(\Diamond A \to (B \,\mathbf{U}\, C))$	$A \to C \,\mathbf{U}\, B$	$A \to \Box(B \,\mathbf{U}\, C)$	(1,5 P)
$\Box(A \to \Diamond(B \,\mathbf{U}\, C \wedge \Diamond B))$	$A \to \Box(B \,\mathbf{U}\, C)$		(1,5 P)
$\Box((A \to B) \wedge (B \,\mathbf{U}\, C))$	$\Box(\Diamond A \to (B \,\mathbf{V}\, C))$	$\Box(A \to B)$	(1 P)
$\Box(A \to B) \wedge \Box(B \,\mathbf{U}\, \neg C)$	$A \to \Diamond B \wedge (\Diamond C \vee \Box B)$		(0,5 P)
keine LTL-Formel			(0 P.)

ii. Ampel:

Immer wenn ein Auto an der Ampel erkannt wird (H), zeigt die Ampel irgendwann grün (G).

*Name: **Vorname** **Familienname***
*Matr.-Nr.: **Matr.-Nr.***

$\Box(H \rightarrow \Diamond G)$	$\Box(H \rightarrow \mathbf{X}\Diamond G)$	$\Box(H \rightarrow \Diamond \mathbf{X} G)$		(2 P.)
$H \rightarrow \Diamond G$	$\Box H(\rightarrow H \,\mathbf{U}\, G)$	$H \rightarrow \mathbf{X} G$	$\Box H \rightarrow \Diamond G$	(1,5 P.)
$H \rightarrow \Box(H \,\mathbf{U}\, G)$	$(\Box H \rightarrow \Diamond \Box G)$	$(\Box H \rightarrow \Box \Diamond G)$		(1 P.)
$\Box(H \,\mathbf{U}\, G)$	$\Box(H \wedge G)$	$\Box(\Diamond H \otimes G)$		(0,5 P)
$\Box(H \mathbf{V} \neg G)$	keine LTL-Formel			(0 P.)