Karlsruher Institut für Technologie

# Klausur Formale Systeme

Fakultät für Informatik

WS 2019/20

Prof. Dr. Bernhard Beckert

28. Februar 2020

**Name:** _____

**Vorname:** _____

**Matrikel-Nr.:** _____

*Die Bearbeitungszeit beträgt 60 Minuten.*

| A1 (12) | A2 (9) | A3 (7) | A4 (8) | A5 (9) | A6 (7) | A7 (8) | Σ (60) |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

**Bewertungstabelle bitte frei lassen!**

**Gesamtpunkte:**

# 1 Zur Einstimmung (4+5+3 = 12 Punkte)

**a.** Seien $p$ ein einstelliges Prädikatensymbol und $f$ ein einstelliges Funktionssymbol.

Geben Sie für folgende prädikatenlogische Formeln – **falls möglich** – Interpretationen $I$ über dem Universum $D = \{a, b\}$ an, und zwar jeweils

- eine Interpretation, in der die Formel **wahr** ist, und
- eine Interpretation, in der die Formel **falsch** ist.

In den Fällen, in denen eine Interpretation mit der gesuchten Eigenschaft **nicht existiert**, geben Sie dies an.

**Hinweis:** Es muss explizit angegeben werden, wenn eine passende Interpretation nicht existiert (schreiben Sie „existiert nicht“ neben den Kasten). Es genügt nicht, die Beschreibung der Interpretation leer zu lassen.

**i.** $\exists x \, \forall y \, (f(x) \doteq y)$

Interpretation, in der die Formel wahr ist:

| $I(f)(a) =$ | $I(f)(b) =$ | **Existiert nicht!** |
|---|---|---|

Interpretation, in der die Formel falsch ist:

| $I(f)(a) = a$ | $I(f)(b) = b$ |
|---|---|

**ii.** $(\exists x \, p(x)) \rightarrow (\exists y \, p(f(y)))$

Interpretation, in der die Formel wahr ist:

| $I(p)(a) = \mathbf{F}$ | $I(p)(b) = \mathbf{F}$ | $I(f)(a) = a$ | $I(f)(b) = b$ |
|---|---|---|---|

Interpretation, in der die Formel falsch ist:

| $I(p)(a) = \mathbf{W}$ | $I(p)(b) = \mathbf{F}$ | $I(f)(a) = b$ | $I(f)(b) = b$ |
|---|---|---|---|

## Fortsetzung 1 Zur Einstimmung

**b.** Geben Sie kurze Antworten zu folgenden Fragen bzw. Aufgaben:

**i.** Zeichnen Sie den reduzierten Shannongraphen, der eine aussagenlogische Tautologie repräsentiert.

**1**

**ii.** Geben Sie eine Substitution $\sigma$ an, für die es **keine** Substitution $\phi$ mit $\phi \circ \sigma = id$ gibt

$\{x \setminus a\}$

**iii.** Wir wollen aussagenlogische Formel in disjunktiver Normalform (DNF) auf Erfüllbarkeit untersuchen. Welche syntaktische Eigenschaft der einzelnen Klauseln muss man dafür überprüfen?

Die Formel ist unerfüllbar gdw. es in jeder Konjunktion ein Atom gibt, das zweimal vorkommt.

**iv.** Seien `m` und `n` zwei Java-Methoden, wobei `m` von `n` aufgerufen wird. Ein JML-Methodenvertrag für `m` enthalte die Vorbedingung „`requires false;`“. Beim Beweis, dass `n` seinen Vertrag erfüllt, wird der Vertrag von `m` als schon bewiesen angenommen und verwendet. Auf welches Problem stößt man dabei?

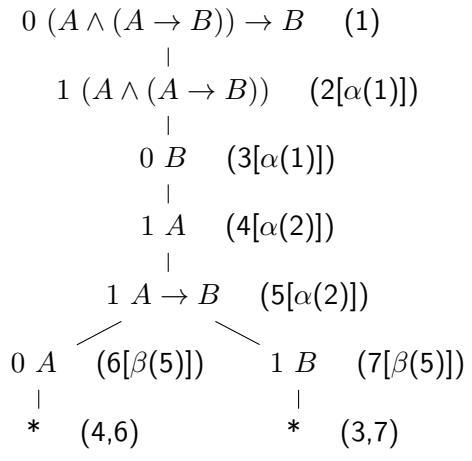
Man kann nicht beweisen, dass die Vorbedingung von `m` gilt.

**v.** Wie überprüft man, ob ein Büchi-Automat eine nicht-leere Sprache akzeptiert?

Man prüft, ob es eine vom Startzustand erreichbare Schleife gibt, die einen Endzustand enthält.

**c.** Zeigen Sie mit Hilfe des **Tableaukalküls** der Aussagenlogik, dass folgende Formel allgemeingültig ist. Notieren Sie dabei:

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



## 2 Theorien ((2+3) + (2+2) = 9 Punkte)

**a.** Wir betrachten die Arithmetik über den natürlichen Zahlen (mit + und $*$).

- Sei $\mathcal{N} = \{\phi \mid (\mathbb{N}, I_{\mathcal{N}}) \models \phi\}$ die Theorie, die durch das Standardmodell $(\mathbb{N}, I_{\mathcal{N}})$ der natürlichen Zahlen definiert ist.
- Sei $\mathcal{PA} = \{\phi \mid PA \models \phi\}$ die Theorie, die durch die Peano-Axiome $PA$ definiert ist.

Diskutieren Sie (kurz) den Unterschied dieser beiden Theorien bzgl. folgender Aspekte:

**i.** Welche der Theorien enthält mehr Formeln, welche hat mehr Modelle?

Es gibt Formeln, die im Standard-Modell der natürlichen Zahlen wahr sind, aber nicht aus den Peano-Axiomen ableitbar (Gödelscher Unvollständigkeitssatz); darum enthält $\mathcal{N}$ mehr Formeln. Umgekehrt hat $\mathcal{PA}$ mehr Modelle, nämlich die Nicht-Standard-Modelle.

**ii.** Sind die beiden Theorien axiomatisierbar? ... entscheidbar? ... rekursiv aufzählbar?

$\mathcal{PA}$ ist axiomatisierbar, nämlich durch die Peano-Axiome $PA$. $\mathcal{PA}$ ist rekursiv aufzählbar, da es vollständige Kalküle für die Prädikatenlogik gibt. $\mathcal{PA}$ ist nicht entscheidbar, das stünde im Widerspruch dazu, dass die Allgemeingültigkeit prädikatenlogischer Formeln nicht entscheidbar ist.
$\mathcal{N}$ hat keine der genannten Eigenschaften; das folgt aus dem Gödelschen Unvollständigkeitssatz.

**b.** Sie arbeiten für eine Firma, die Motoren mit der dafür notwendigen Steuerungssoftware herstellt. Sie sollen Lizenzen für SMT-Solver erwerben, die die Entwicklungsingenieure der Firma nutzbringend verwenden können. Auf dem Markt gibt es vier SMT-Solver, die jeweils eine Theorie besonders gut unterstützen: (1) Arithmetik ganzer Zahlen, (2) Arithmetik reeller Zahlen, (3) Fließkomma-Arithmetik und (4) Arithmetik über Bitvektoren. Leider kann sich die Firma nur zwei dieser SMT-Solver leisten. Welche zwei wählen Sie und warum?

**Hinweis**: Es gibt für jeden der vier SMT-Solver gute Argumente.

**i.** SMT-Solver 1:

**Arithmetik ganzer Zahlen:** Die Steuerungssoftware rechnet mit Binärzahlen, was sich auf die Arithmetik ganzer Zahlen abbilden lässt. Allerdings ist die Theorie unentscheidbar.
**Arithmetik reller Zahlen:** Die physikalischen Vorgänge in einem Motor können mit reellwertigen Zahlen modelliert werden. Die Theorie ist entscheidbar.
**Fließkomma-Arithmetik:** Da die Steuerungssoftware Berechnungen mit Bezug auf physikalische Software anstellen muss, verwendet sie Fließkomma-Zahlen. Diese Theorie ist endlich und damit entscheidbar; allerdings in der Praxis schwer handhabbar.
**Bitvektoren:** Die einfache/endliche Variante der Arithmetik ganzer Zahlen, die zur Modellierung der in der Software verwendeten 16- oder 32-bit-Zahlen ausreicht. Diese Theorie ist entscheidbar.

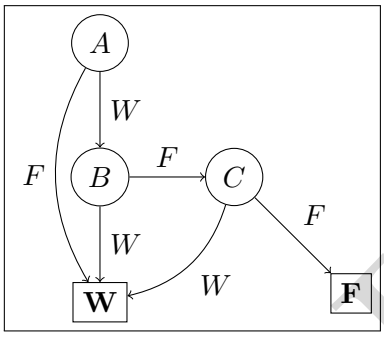
**ii.** SMT-Solver 2:

s.o.

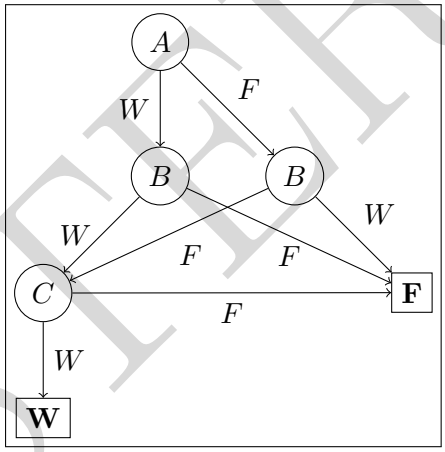
## 3 Shannongraphen (3+4 = 7 Punkte)

Zeichnen Sie zu jeder der folgenden aussagenlogischen Formeln den reduzierten Shannongraphen. Die Variablenordnung ist dabei $A < B < C$.

**a.** $(A \to B) \vee C$




**b.** $(A \leftrightarrow B) \wedge C$



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

Gegeben sei die prädikatenlogische Signatur $\Sigma = (\{sat\}, \{\emptyset, union, calc\}, \alpha)$. Sie enthält das Prädikatensymbol $sat(\cdot)$, sowie die Funktionssymbole $\emptyset$, $union(\cdot, \cdot)$ und $calc(\cdot)$.

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

- das Universum $D$ eine Menge aussagenlogischer Formelmengen ist,
- das Prädikat $sat(x)$ genau dann wahr ist, wenn die Formelmenge $x$ erfüllbar ist,
- die Konstante $\emptyset$ die leere Formelmenge repräsentiert,
- die Funktion $calc(x)$ eine durch die Anwendung des Kalküls $K$ aus $x$ erhaltene Formelmenge zurückliefert,
- die Funktion $union(x_1, x_2)$ die Vereinigung der Formelmengen $x_1$ und $x_2$ zurückliefert.

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

**a.** Die leere Formelmenge ist erfüllbar.

$\boxed{sat(\emptyset)}$

---

**b.** Beim Anwenden des Kalküls $K$ auf eine beliebige Formelmenge erhält man eine erfüllbarkeitsäquivalente Formelmenge.

$\boxed{\forall x_1 \left(sat(x_1) \leftrightarrow sat(calc(x_1))\right)}$

---

**c.** Die Vereinigung zweier erfüllbarer Formelmengen kann unerfüllbar sein.

$\boxed{\exists x_1 \, \exists x_2 \left(sat(x_1) \wedge sat(x_2) \wedge \neg sat(union(x_1, x_2))\right)}$

---

**d.** Beim Anwenden des Kalküls $K$ auf eine erfüllbare Formelmenge erhält man die leere Formelmenge.

$\boxed{\forall x_1 \left(sat(x_1) \rightarrow (calc(x_1) \doteq \emptyset)\right)}$

---

## 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 die die Resolutionsregel angewandt wurde, sowie **die verwendete Substitution**.

**Signatur:** $p, q$ sind Prädikate; $c$ ist eine Konstante; $f$ ist eine Funktion; $x_1, x_2, x_3, x_4, x_5$ sind Variablen.

**Hinweis**: Mit der allgemeinen Resolutionsregel können die Klauseln (2) und (3) so resolviert werden, dass eine Einerklausel entsteht.

(1) $\{q(x_1),\ q(f(x_1))\}$ (2) $\{\neg q(x_2),\ \neg q(f(x_2))\}$ (3) $\{q(x_3),\ q(c))\}$

(4) $\{\neg q(f(f(c))),\ \neg p(x_4)\}$ (5) $\{p(x_5), \neg q(f(x_5))\}$

(7) $\{\neg q(f(f(c))),\ \neg q(f(x_4))\}$ aus $4, 5$ mit $\sigma = \{x_5 \setminus x_4\}$

(8) $\{q(f(c))\}$ aus $1, 7$ mit $\sigma = \{x_1 \setminus f(c),\ x_4 \setminus f(c)\}$

(9) $\{\neg q(f(c))\}$ aus $2, 3$ mit $\sigma = \{x_2 \setminus c,\ x_3 \setminus c\}$

(10) $\square$ aus $8, 9$ mit $\sigma = id$.

# 6 Spezifikation mit der Java Modeling Language

**(3+4 = 7 Punkte)**

**a.** Geben Sie die Bedeutung der Nachbedingung (**ensures-Klauseln**) in dem folgenden JML-Methodenvertrag in natürlicher Sprache wieder.

```
public class A {
    public int[] s;
    public int[] t;

    /*@ normal_behaviour
      @  requires t != null;
      @  assignable t;
      @  ensures t != null && t.length == \old(t.length);
      @  ensures (\forall int i; 0 <= i && i < t.length;
      @              i < s.length ==> t[i] == s[i]);
      @  ensures (\forall int i; 0 <= i && i < t.length;
      @              s.length <= i ==> t[i] == p);
      @*/
    public void m(int p) { ... }
}
```

Wenn `m` aufgerufen wird, terminiert `m` und und es gilt nach `m`'s Ausführung:

- Das Array `t` ist ungleich `null` und hat genau so viele Elemente wie im Vorzustand.
- Das Array `t` enthält an seinen Positionen die Einträge des Arrays `s` mit gleichem Positionsindex von `0` bis zum Minimum aus `s.length - 1` und `t.length - 1`, die Reihenfolge bleibt dabei unverändert.
- Die übrigen Positionen in `t`, insofern die Länge von `t` größer als die Länge von `s` ist, werden mit dem Parameter `p` überschrieben.

## Fortsetzung 6 Spezifikation mit der Java Modeling Language

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

Die Methode terminiert, wirft keine Exception und erfüllt folgende Nachbedingungen:

- Die Methode gibt ein von `null` verschiedenes `int`-Array zurück, das entweder genau so viele Elemente wie das Array `s` oder genau so viele Elemente wie das Array `t` enthält. Das zurückgegebene Array enthält aber nicht mehr Elemente als `s` und nicht mehr Elemente als `t`.
- Dabei enthält das zurückgegebene Array an jeder Stelle den Absolutwert der Differenz zwischen den Werten aus `s` und `t` an der jeweils gleichen Indexposition.

```
public class A {
    public int[] s;
    public int[] t;
    /*@
      @
      @
      @
      @
      @
      @
      @
      @
      @
      @
      @
      @
      @
      @*/
    public int[] f() { ... }
}
```

```
/*@ public normal_behaviour
  @ ensures \result.length == s.length || \result.length == t.length;
  @ ensures \result.length <= s.length && \result.length <= t.length;
  @ ensures (\forall int i; 0 <= i && i < \result.length;
  @             s[i] <= t[i] ==> \result[i] == t[i] - s[i]);
  @ ensures (\forall int i; 0 <= i && i < \result.length;
  @             s[i] > t[i] ==> \result[i] == s[i] - t[i]);
  @*/
```

# 7 Lineare Temporale Logik (LTL) und Büchi-Automaten

**((2+1)+5 = 8 Punkte)**

**a.** Release-Operator

**i.** Formen Sie die LTL-Formel $\Box a$ so um, dass Sie eine äquivalente LTL-Formel erhalten, die nur den Release-Operator ($a \mathbf{V} b$) verwendet. Verwenden Sie dazu die Tautologien aus der Vorlesung.
**Hinweis:** $a \,\mathbf{V}\, b \quad \equiv \quad \neg(\neg a \,\mathbf{U}\, \neg b)$

$\Box a \quad \equiv \quad \boxed{\Box a \equiv \neg\Diamond\neg a \equiv \neg(true \,\mathbf{U}\, \neg a) \equiv \neg(\neg false \,\mathbf{U}\, \neg a) \equiv false \,\mathbf{V}\, a}$

**ii.** Bildet der Release-Operator eine Basis für die LTL-Operatoren ($\Box, \Diamond, \mathbf{U}, \mathbf{X}$)? Begründen Sie!

Nein. Der Next-Operator ist nicht darstellbar, alle anderen schon (siehe oben).

**b.** Geben Sie einen nicht-deterministischen Büchi-Automaten an, dessen akzeptierte Sprache den Modellen ($\omega$-Wörtern) der LTL-Formel

$(\Diamond a) \rightarrow ((\neg b) \,\mathbf{U}\, a)$

über der Signatur $\Sigma = \{a, b\}$ entspricht.

Für das Vokabular $V = \mathbb{P}(\Sigma)$ (Potenzmenge von $\Sigma$) werden die folgenden, aus der Vorlesung bekannten Abkürzungen definiert:

$A = \{M \in V \mid a \in M\} \subset V$ $\qquad$ $B = \{M \in V \mid b \in M\} \subset V$
$\bar{A} = \{M \in V \mid a \notin M\} \subset V$ $\qquad$ $\bar{B} = \{M \in V \mid b \notin M\} \subset V$

Umformungstrick: $(\Diamond a) \rightarrow (\neg b) \,\mathbf{U}\, a \equiv (G\neg a) \vee (\neg b) \,\mathbf{U}\, a$