Karlsruher Institut für Technologie

Name: _____

Vorname: _____

Matrikel-Nr.: _____

Klausurnummer:

_____

Klausur Formale Systeme

Fakultät für Informatik

WS 2024/25

Prof. Dr. Bernhard Beckert

28. Februar 2025

Die Bearbeitungszeit beträgt 60 Minuten.

A1 (7)	A2 (11)	A3 (6,5)	A4 (7)	A5 (10)	A6 (10)	A7 (8,5)	Σ (60)

Bewertungstabelle bitte frei lassen!

Gesamtpunkte:

1 Zur Einstimmung (2 + 2 + 2 + 1 = 7 Punkte)

a. Transformieren Sie die folgende prädikatenlogische Formel in eine äquivalente Formel in pränexer Normalform.

$(\exists x\, q(x)) \;\rightarrow\; (\forall x\, p(f(x)))$

$\forall x_0\ \forall x_1\ \ q(x_0) \rightarrow p(f(x_1))$
oder
$\forall x_1\ \forall x_0\ \ q(x_0) \rightarrow p(f(x_1))$

b. Transformieren Sie die folgende prädikatenlogische Formel in Skolem-Normalform. Notieren Sie, welche Symbole in **Ihrer** Formel (also in Ihrem Ergebnis) Prädikatssymbole, Funktionssymbole und Variablen sind.

$\exists w\ \forall x\ \exists y\ \exists z\ \ (\neg p(w, x) \vee q(x, y, z))$

Die Lösung lautet:
$\forall x\ \ (\neg p(c_1, x) \vee q(x, f(x), g(x)))$
x ist eine Variable; c_1, f und g sind Funktionssymbole; p und q sind Prädikate.

c. Sie wissen aus der Vorlesung, dass es einen polynomiellen Lösungsalgorithmus für das 2-SAT-Problem gibt. Ein Kommilitone erzählt Ihnen, dass er diesen polynomiellen Algorithmus auf 3-SAT erweitert hat. Welche Implikation hätte es, wenn sich der Algorithmus als korrekt herausstellen würde? Begründen Sie.

Dies würde implizieren, dass P=NP gilt.
Der Grund hierfür ist, dass das 3-SAT Problem NP-vollständig ist. Dies liegt darin begründet, dass sich beliebige SAT-Probleme auf 3-SAT reduzieren lassen, indem man längere Klauseln durch die Einführung neuer Variablen auf Klauseln der Länge 3 reduziert.

d. Erklären Sie den Zusammenhang zwischen dem Reduktionsschritt für Einerklauseln (in der Notation der Vorlesung $\mathrm{red}_{\{L\}}(S)$; auch Unit Propagation genannt) im DPLL-Algorithmus und der Resolutionsregel im aussagenlogischen Resolutionskalkül.

Hinweis: Beschränken Sie Ihre Erläuterung auf die Behandlung von Klauseln, die vom Reduktionsschritt *modifiziert* werden. Den Fall, in dem der Reduktionsschritt eine Klausel *entfernt*, können Sie ignorieren.

Der Reduktionsschritt nutzt eine Einer-Klausel mit Literal l, um aus allen anderen Klauseln $\{\overline{l}\} \cup C$ das Komplement $\overline{l}$ zu entfernen. Das Ergebnis dieser Operation entspricht der Anwendung der Resolutionsregel auf $\{l\}$ und $\{\overline{l}\} \cup C$.

2 Para-konsistente Logik (3 + 4 + 4 = 11 Punkte)

In der klassischen Logik führt ein Widerspruch dazu, dass alles beweisbar wird: $\{P, \neg P\} \models Q$ für alle P, Q. Para-konsistente Logiken sind nicht-klassische Logiken, die dieses Phänomen vermeiden. Sie ermöglichen es, Widersprüche auf eine kontrollierte Art zu handhaben.

Wir betrachten nun eine para-konsistente dreiwertige Aussagenlogik, die neben den klassischen Wahrheitswerten

T (true) und F (false)

einen weiteren Wahrheitswert hat, nämlich

B (both).

Die Intuiton von B ist „sowohl T als auch F“. Die Semantik der Operatoren $\wedge, \vee, \neg$ in dieser Logik finden Sie rechts.

$\wedge$	T	B	F
T	T	B	F
B	B	B	F
F	F	F	F

$\vee$	T	B	F
T	T	T	T
B	T	B	B
F	T	B	F

X	$\neg X$
T	F
F	T
B	B

a. Nehmen Sie an, die Variable X hat den Wahrheitswert B (also $\text{val}(X) = B$) und die Variable Y hat den Wahrheitswert F (also $\text{val}(Y) = F$). Welchen Wahrheitswert sollte man dann $X \to Y$ zuweisen? Begründen Sie, warum Ihre Antwort der Intuition entspricht.

Man sollte den Wahrheitswert B zuweisen, denn $F \to F$ hat den Wahrheitswert T und $T \to F$ hat den Wahrheitswert F, so dass $B \to F$ beide Wahrheitswerte hat. (Der Bezug auf die Äquivalenz $X \to Y \equiv \neg X \vee Y$ genügt als Begründung nicht, weil dies nicht auf die Intuition von B als „sowohl als auch“ Bezug nimmt.)

b. Wir definieren, dass eine Formel G in einem Modell M wahr ist, wenn sie dort den Wahrheitswert T oder den Wahrheitswert B hat.

Begründen Sie, warum dann – wie gewünscht –

$$\{P, \neg P\} \models Q$$

nicht gilt.

Seien P, Q aussagenlogische Variablen und M ein Modell, in dem P den Wahrheitswert B hat und Q den Wahrheitswert F. Dann sind P und $\neg P$ beide in M wahr. Also ist M ein Modell der Formelmenge $\{P, \neg P\}$. M ist aber kein Modell von Q. $\{P, \neg P\} \models Q$ gilt also nicht, denn das würde erfordern, dass Q in jedem Modell von $\{P, \neg P\}$ wahr ist.

c. Wir definieren, dass eine Formel G eine Tautologie in dieser Logik ist, wenn sie in jeder Interpretation den Wahrheitswert T oder den Wahrheitswert B hat.

Begründen Sie, warum jede Tautologie der para-konsistenten Logik auch eine Tautologie der klassischen zweiwertigen Logik ist.

Die Definition der logischen Operatoren stimmt in der para-konsistenten Logik mit der klassischen Logik überein für alle Fälle, in denen die Operanden einen der Werte T, F (und nicht B) haben. Wenn also G eine para-konsistente Tautologie ist, kann sie in Interpretationen, in denen alle Variablen mit T, F belegt sind – wie es in Modellen der klassischen Logik immer der Fall ist – nur die Werte T, F (und nicht B) annehmen. Zugleich ist sie aber eine para-konsistente Tautologie und kann daher nicht den Wert false annehmen. Also ist sie in allen Modellen der klassischen Logik den Wahrheitswert T und ist eine Tautologie der klassischen Logik. (Punktabzug, wenn der Hinweis darauf fehlt, dass die Beschränkung der para-konsistenten Logik auf T, F mit den klassischen Operatoren übereinstimm.)

3 Markierungsalgorithmus für Hornformeln

(2 + 4,5 = 6,5 Punkte)

a. Sind die nachfolgenden Formeln jeweils Hornformeln? Wenn nein, begründen Sie!

	Formel	Hornformel	keine Hornformel
(i)	$P_1 \wedge P_2$	☒	☐
(ii)	$P_1 \vee P_2$	☐	☒
(iii)	$((\neg P_0 \wedge P_1) \rightarrow P_3) \wedge P_2$	☐	☒
(iv)	$(\neg P_0 \wedge P_1 \wedge \neg P_3 \wedge \neg P_4)$	☒	☐

Begründungen:

(ii): $P_1 \vee P_2$ ist eine einzige Klausel mit mehr als einem positiven Literal.
(iii): $(\neg P_0 \wedge P_1 \rightarrow P_3) \equiv (P_0 \vee \neg P_1 \vee P_3)$ und hat damit ebenfalls mehr als ein positives Literal.

b. Überprüfen Sie die folgenden Hornformeln auf Erfüllbarkeit. Benutzen Sie den Markierungsalgorithmus (auch „Erfüllbarkeitstest für Hornformeln“) aus der Vorlesung. **Unterstreichen** Sie dazu die zu markierenden Literale in der Formel **und** geben Sie unter Schritt n an, **welche(s) Literal(e) im n-ten Schritt markiert** wurde(n). Geben Sie **zudem** ein **Modell** an **oder** benennen Sie die **Hornformel**, aufgrund derer der Algorithmus mit „unerfüllbar“ abbricht! Bei einem Modell ist für jedes Atom explizit anzugeben, ob es zu wahr oder falsch auswertet.

Hinweis: Es sind nicht immer 4 Schritte nötig.

i. $P_1 \wedge (\neg P_1 \vee \neg P_2 \vee \neg P_3) \wedge (\neg P_1 \vee P_2)$

Schritt 1:	Schritt 2:	Schritt 3:	Schritt 4:
P_1	P_2		

Ergebnis: Erfüllbar mit $I(P_1) = I(P_2) = W$ und $I(P_3) = F$.

ii. $(\neg P_1 \vee \neg P_2) \wedge P_2 \wedge (\neg P_2 \vee P_3) \wedge (\neg P_3 \vee P_1)$

Schritt 1:	Schritt 2:	Schritt 3:	Schritt 4:
P_2	P_3	P_1	

Ergebnis: Nicht erfüllbar. Konfliktklausel $(\neg P_1 \vee \neg P_2) \equiv P_1 \wedge P_2 \rightarrow \mathbf{0}$

iii. $(P_1 \rightarrow P_2) \wedge (P_2 \rightarrow P_3) \wedge ((P_1 \wedge P_3) \rightarrow P_4)$

Schritt 1:	Schritt 2:	Schritt 3:	Schritt 4:

Ergebnis: Erfüllbar, keine Fakten. Belegung: $I(P_1) = I(P_2) = I(P_3) = I(P_4) = F$

4 Formalisieren in Prädikatenlogik (PL1)

(1,5 + 1,5 + 2 + 2 = 7 Punkte)

Im Folgenden soll mithilfe von Prädikatenlogik ein vereinfachtes Public-Key-Kryptosystem zur Ver- und Entschlüsselung formalisiert werden. Hierzu sei die prädikatenlogische Signatur $\Sigma = (\{enc, dec\}, \{keypair\}, \alpha)$ gegeben. Sie enthält das zweistellige Prädikatssymbol $keypair(\cdot,\cdot)$ sowie die zweistelligen Funktionssymbole $enc(\cdot,\cdot)$ und $dec(\cdot,\cdot)$.

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

- das Universum D eine Menge von Strings ist, die als Texte (Klartext oder Chiffrat) sowie als Schlüssel (öffentlich oder geheim) aufgefasst werden können,
- $keypair(x, y)$ genau dann wahr ist, wenn x der passende öffentliche Schlüssel zum geheimen Schlüssel y ist,
- $enc(x, y)$ das Chiffrat (also die Verschlüsselung) von Text x mit y als Schlüssel ist, und
- $dec(x, y)$ der Text ist, den man erhält, wenn man den Text x als Chiffrat auffasst und mit y als Schlüssel entschlüsselt.

Hinweis: Ist y kein (passender) Schlüssel, haben *enc* und *dec* trotzdem ein Ergebnis. Das Ergebnis einer Anwendung von *enc* nennen wir *Chiffrat*, eine Anwendung von *dec* eine *Entschlüsselung*.

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

a. Für einen Klartext und ein Chiffrat kann es nur höchstens einen Schlüssel geben, sodass die Verschlüsselung dieses Klartextes zu diesem Chiffrat führt.

$$\forall m\ \forall k\ \forall l \Big(\neg k \doteq l \rightarrow \neg enc(m,k) \doteq enc(m,l) \Big)$$

b. Der öffentliche Schlüssel eines Schlüsselpaares passt zu keinem anderen geheimen Schlüssel und der geheime Schlüssel dieses Schlüsselpaares passt zu keinem anderen öffentlichen Schlüssel.

$$\forall k\ \forall l\ \forall s\ \forall t \Big(keypair(k,s) \land \neg k \doteq l \land \neg s \doteq t \rightarrow \neg keypair(k,t) \land \neg keypair(l,s) \Big)$$

c. Wenn ein Chiffrat zusätzlich mit einem weiteren (öffentlichen) Schlüssel verschlüsselt wird, führt die Hintereinanderausführung der Entschlüsselungen mit den beiden passenden (geheimen) Schlüsseln in jeder Reihenfolge der beiden Entschlüsselungen zum gleichen Ergebnis.

$$\forall k\ \forall l\ \forall s\ \forall t\ \forall m\ \forall c$$
$$\Big(keypair(k,s) \land keypair(l,t) \land enc(enc(m,k),l) \doteq c \rightarrow dec(dec(c,s),t) \doteq dec(dec(c,t),s) \Big)$$

d. Zu jedem Chiffrat gibt es einen Schlüssel, mit dem die Entschlüsselung zum ursprünglichen Text führt und der ein Paar mit dem Schlüssel bildet, der zur Verschlüsselung verwendet wurde. Eine Entschlüsselung des Chiffrats führt aber mit keinem anderen Schlüssel zum ursprünglichen Text.

$$\forall m\ \forall k\ \forall c \Big(c \doteq enc(m,k) \rightarrow \exists s \Big(keypair(k,s) \land dec(c,s) \doteq m \land \forall t \big(\neg s \doteq t \rightarrow \neg dec(c,t) \doteq m \big) \Big) \Big)$$

5 Resolutionskalkül (2 + 3 + 5 = 10 Punkte)

a. Geben Sie für die folgende Klauselmenge ein Modell (D, I) an. Nutzen Sie als Universum des Modells die Menge der ganzen Zahlen, also $D = \mathbb{Z}$.

$\{\ \{r(x,y), r(y,x)\},\ \{\neg r(x, f(x))\}\ \}$

$r(x,y)$ ist wahr gdw. $x \leq y$
$f(x)$ bildet auf $x - 1$ ab.

b. Beweisen Sie die Unerfüllbarkeit der folgenden Klauselmenge mithilfe des Resolutionskalküls. Geben Sie alle notwendigen Substitutionen an.

$\{\ \{r(x,y), r(y,z)\},\ \{\neg r(x, f(x))\}\ \}$

Hinweis: Beachten Sie die unterschiedlichen Variablen des Prädikats r im Vergleich zur Aufgabe a (x vs. z).

$$\dfrac{\dfrac{r(x,y) \vee r(y,z) \qquad \neg r(w, f(w))}{r(f(u), z)} \qquad \neg r(w, f(w))}{\square}$$

Erste Substitution: $\sigma = \{x/u,\ y/f(u),\ w/u\}$
Zweite Substitution: $\sigma = \{w/f(u), z/f(f(u))\}$

c. Beweisen Sie die Unerfüllbarkeit der folgenden Klauselmenge mithilfe des Resolutionskalküls. Geben Sie alle notwendigen Substitutionen an.

$$\{\ \{\neg r(x,y), r(y,x)\}\ ,\ \{\neg r(z,u), \neg r(u,w), r(z,w)\}\ ,\ \{r(v,f(v))\}\ ,\ \{\neg r(c,c)\}\}$$

Beachten Sie, dass es sich bei u, v, w, x, y, z um Variablen handelt, während c eine Konstante und f ein Funktionssymbol ist.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{r(v,f(v)) \qquad \neg r(x,y) \vee r(y,x)}{r(f(v),v)} \qquad \neg r(z,u) \vee \neg r(u,w) \vee r(z,w)}{\neg r(z,f(v)) \vee r(z,v)} \qquad r(v,f(v))}{r(v,v)} \qquad \neg r(c,c)}{\square}$$

Erste Substitution: $\sigma = \{x/v,\ y/f(v)\}$
Zweite Substitution: $\sigma = \{u/f(v),\ w/v\}$
Dritte Substitution: $\sigma = \{z/v\}$
Vierte Substitution: $\sigma = \{v/c\}$

6 Spezifikation mit der Java Modeling Language (JML)

(5 + (2 + 3) = 10 Punkte)

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

```
public final class A {
    /*@ normal_behavior
      @  requires (\forall int i; 0 <= i && i < a.length;
      @              (\forall int j; 0 <= j && j < a.length;
      @                              i != j => a[i] != a[j]));
      @
      @  ensures \result.length == (\sum int k; 0 <= k && k < a.length;
      @                                         a[k] % 2 == 0 ? 1 : 0);
      @  ensures (\forall int l; 0 <= l && l < \result.length;
      @                             \result[l] % 2 == 0
      @                          && (\exists int m; 0 <= m && m < a.length;
      @                                             \result[l] == a[m]));
      @  assignable \nothing;
      @*/
    static int[] m(int[] a) { ... }
}
```

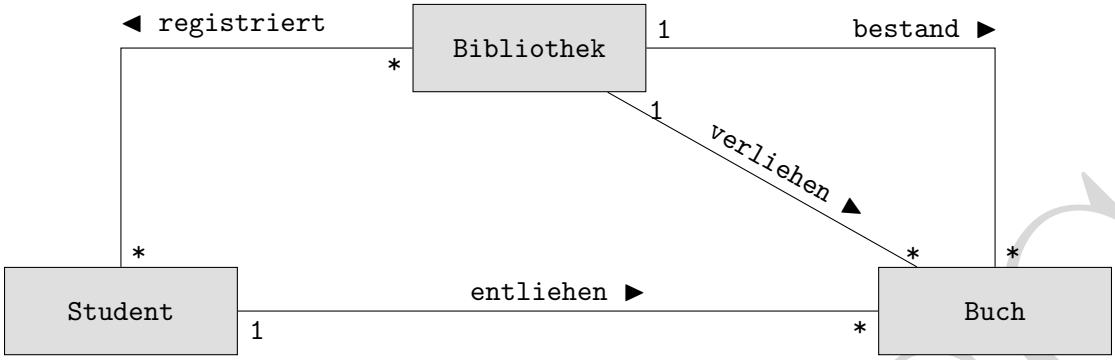
Hinweis: `\sum` wird hier verwendet um Elemente abzuzählen.

Wenn `m` mit einem int-Array aufgerufen wird, dessen Einträge paarweise verschieden sind, terminiert die Methode ohne Exception und ändert keine (vor Aufruf der Methode existierenden) Speicherstellen auf dem Heap. Der Rückgabewert der Methode ist dabei ein Array, dessen Länge der Anzahl der geraden Einträge im Eingabearray entspricht. Dabei gilt für jeden Eintrag des Ergebnisarrays, dass er gerade ist und schon im Eingabearray existiert.
Anmerkung: Der Vertrag stellt genau genommen nicht sicher, dass jede Zahl im Ergebnisarray nur einmal enthalten ist. Daher erlaubt der Vertrag auch, dass z.B. k-mal die erste gerade Zahl der Eingabe zurückgegeben wird (wobei k die Anzahl gerader Elemente der Eingabe ist).
Bei der Korrektur geben Lösungen der folgenden Art aber trotzdem volle Punktzahl, weil das die ursprüngliche Intention der Aufgabe war und man dieses Detail sehr leicht übersehen konnte:

- "Die Methode filtert alle ungeraden Elemente aus dem Eingabearray heraus."
- "Die Methode gibt ein Array zurück, das die/alle geraden Elemente des Eingabearrays enthält."

Außerdem macht der Vertrag keinerlei Aussage über die Reihenfolge der Elemente im Ergebnisarray.

Fortsetzung 6 Spezifikation mit der Java Modeling Language



b. Unten sehen Sie den Java-Code zum oben abgebildeten UML-Klassendiagramm. Geben Sie JML-Klasseninvarianten für die Klasse `Bibliothek` an, die die Aussagen **i.** und **ii.** formalisieren.

Hinweise: Sie können annehmen, dass es zu jedem Buch nur ein Java-Objekt gibt, insbesondere können Bücher also direkt mit `==` verglichen werden.

i. „Jedes von der Bibliothek verliehene Buch ist im Bestand enthalten."

ii. „Jedes von der Bibliothek verliehene Buch wurde von einem der bei dieser Bibliothek registrierten Studenten entliehen."

```
class Student {
  Buch[] entliehen;
}

class Buch {
  ...
}

class Bibliothek {
  Student[] registriert;
  Buch[] bestand;
  Buch[] verliehen;
   /*@ invariant (\forall int i; 0 <= i && i < verliehen.length;
     @             (\exists int j; 0 <= j && j < bestand.length;
     @                           verliehen[i] == bestand[j]));
     @*/


  /*@ invariant (\forall int i; 0 <= i && i < verliehen.length;
    @             (\exists int j; 0 <= j && j < registriert.length;
    @               (\exists int k; 0 <= k && k < registriert[j].entliehen.length;
    @                         registriert[j].entliehen[k] == verliehen[i])));
    @*/
}
```

7 Lineare Temporale Logik (LTL) (3 * 1,5 + 4 = 8,5 Punkte)

Im Folgenden soll mithilfe von LTL das Verhalten einer CI/CD-Pipeline („Continuous Integration / Continuos Delivery“) in der Softwareentwicklung beschrieben werden. Die hier betrachtete Pipeline hat drei Phasen: Kompilierung („Build“), Testen („Test“) und Auslieferung („Deploy“). Die Phasen werden jeweils mit drei Variablen beschrieben. Diese sagen aus, ob die jeweilige Phase derzeit aktiv ist oder ob sie – erfolgreich oder nicht – abgeschlossen wurde. Dazu wird die Signatur $\Sigma = \{\ b,\ b_s,\ b_f,\ \ t,\ t_s,\ t_f,\ \ d, d_s, d_f\ \}$ verwendet:

- b ist genau dann wahr, wenn sich die Pipeline in der Phase „Build“ befindet.
- b_s ist genau dann wahr, wenn die Phase „Build“ erfolgreich abgeschlossen wurde.
- b_f ist genau dann wahr, wenn die Phase „Build“ fehlgeschlagen ist.

Die Bedeutung der Variablen für die beiden anderen Phasen ergibt sich entsprechend.

a. Übersetzen Sie die folgende Aussage in LTL:

Die Deploy-Phase kann erst beginnen, nachdem die Build-Phase und die Test-Phase erfolgreich abgeschlossen wurden.

$\neg d\ \mathbf{U_W}\ (b_s \wedge t_s)$

b. Übersetzen Sie die folgende LTL-Formel in natürliche Sprache:

$\Box((b \wedge \mathbf{X}\neg b) \rightarrow \mathbf{X}\Box\neg b)$

Nachdem die Build-Phase beendet ist, wird sich die Pipeline danach nie mehr in der Build-Phase befinden.

c. Erstellen Sie eine weitere sinnvolle Eigenschaft der Pipeline in LTL und geben Sie sie auch in natürlicher Sprache wieder. Die Regel muss mindestens zwei verschiedene Symbole aus Σ und mindestens einen temporalen LTL-Operator enthalten.

$\Box(t \rightarrow t\ \mathbf{U}\ (t_s \vee t_f))$
Wenn sich die Pipeline in der Test-Phase befindet, bleibt sie darin, bis die Phase entweder fehlgeschlagen oder erfolgreich abgeschlossen ist.

d. In den Anforderungen an die Pipeline steht die folgende Formel:

$F = \quad (\neg d \,\mathbf{U}\, t_s) \quad \wedge \quad \Box(t_s \rightarrow \Box t_s) \quad \wedge \quad \Box(t_s \rightarrow (d \vee \mathbf{X} d))$

Geben Sie einen nicht-deterministischen Büchi-Automaten an, dessen akzeptierte Sprache der Formel F entspricht.

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

$$T_s = \{X \in V \mid t_s \in X\} \subset V \qquad D = \{X \in V \mid d \in X\} \subset V$$
$$\overline{T}_s = \{X \in V \mid t_s \notin X\} \subset V \qquad \overline{D} = \{X \in V \mid d \notin X\} \subset V$$

Die Abkürzungen für die restlichen Symbole aus Σ ergeben sich analog, werden zur Bearbeitung dieser Aufgabe aber nicht benötigt.

Sie dürfen an den Kanten des Automaten auch Ausdrücke wie $T_s \cap D$ verwenden.