Karlsruher Institut für Technologie

Name: _____

Vorname: _____

Matrikel-Nr.: _____

Klausur Formale Systeme

Fakultät für Informatik

WS 2023/24

Prof. Dr. Bernhard Beckert

28. Februar 2024

Die Bearbeitungszeit beträgt 60 Minuten.

A1 (10)	A2 (9)	A3 (7)	A4 (7)	A5 (10)	A6 (9)	A7 (8)	Σ (60)

Bewertungstabelle bitte frei lassen!

Gesamtpunkte:

1 Zur Einstimmung ((2 + 1 + 1 + 2) + 4 = 10 Punkte)

a. Geben Sie kurze Antworten zu folgenden Fragen bzw. Aufgaben:

i. Gegeben sei die Signatur $\Sigma = (\{p, q\}, \{c, d\}, \alpha)$ mit Stelligkeiten $\alpha(p) = \alpha(q) = 1$ und $\alpha(c) = \alpha(d) = 0$.
Geben Sie ein Herbrand-Modell (D, I) über Σ an für die Formel $(\forall x\, p(x)) \wedge \neg q(c)$.

$D = \{c, d\}$ (1 Pt)
$I(c) = c$
$I(d) = d$

$I(p)(c) = \mathbf{W}$ (1 Pt)
$I(p)(d) = \mathbf{W}$

$I(q)(c) = \mathbf{F}$
$I(q)(d) = \textit{beliebig}$

ii. Nennen Sie jeweils eine Theorie in Prädikatenlogik erster Stufe, deren Erfüllbarkeitsproblem entscheidbar bzw. nicht entscheidbar ist:

Entscheidbar: Bitvektoren, reelle Arithmetik, Lineare Arithmetik (0,5 Pt)
Nicht entscheidbar: Ganzzahl Arithmetik, UF, UF+Gleichheit (0,5 Pt)

iii. Sei F eine aussagenlogische Tautologie über den Variablen A, B, C. Wie viele unterschiedliche reduzierte Shannongraphen, die F darstellen, gibt es?

1

iv. Gegeben sei eine aussagenlogische Klauselmenge mit m Klauseln über n Variablen, von denen k Klauseln Einerklauseln sind. Wie viele Fallunterscheidungen benötigt DPLL maximal, um die Unerfüllbarkeit zu beweisen? Geben Sie die genaue Schranke an (keine O-Notation) und begründen Sie diese.

$2^{n-k-1} - 1$ (1 Pt)
Die Belegung der ersten k Variablen wird durch Reduktion auf den Einerklauseln determiniert. Damit müssen höchstens noch $n - k - 1$ Variablen Fallunterscheidung durchgeführt werden (für die letzte Variable braucht man keine Fallunterscheidung mehr). Im Worst-Case passieren diese Fallunterscheidungen in jedem Teilbaum (d.h. bspw. wird nach der Fallunterscheidung für A in jedem Zweig eine Fallunterscheidung für B durchgeführt). Dies ergibt einen Baum der Höhe $n - k - 1$ mit $2^{n-k-1} - 1$ inneren Knoten die jeweils eine Fallunterscheidung repräsentieren. (1 Pt)

Fortsetzung 1 Zur Einstimmung

b. Geben Sie korrekte und vollständige Tableaukalkülregeln für den Shannon-Operator sh an. Denken Sie daran, die Regeln für beide Vorzeichen anzugeben.

Hinweis: Per Definition von sh gilt $\mathrm{sh}(\mathbf{W}, P, Q) \equiv Q$ und $\mathrm{sh}(\mathbf{F}, P, Q) \equiv P$.

$\mathbf{1}sh(C, A, B)$	
$\mathbf{0}C$	$\mathbf{1}B$
$\mathbf{1}A$	$\mathbf{1}C$

$\mathbf{0}sh(C, A, B)$	
$\mathbf{0}C$	$\mathbf{0}B$
$\mathbf{0}A$	$\mathbf{1}C$

oder:

$\mathbf{0}sh(C, A, B)$		
$\mathbf{1}C$	$\mathbf{0}C$	$\mathbf{0}A$
$\mathbf{0}B$	$\mathbf{0}A$	$\mathbf{0}B$

2 Unsat Core (2 + 2 + 3 + 2 = 9 Punkte)

Das Konzept des *Unsat Core* ist wie folgt definiert:

> Sei U eine unerfüllbare Formelmenge. Eine Teilmenge $C \subseteq U$ ist ein *Unsat Core* (von U) genau dann, wenn:
>
> 1. C ist unerfüllbar,
> 2. jede echte Teilmenge $C' \subsetneq C$ ist erfüllbar.

Hinweis: Diese Definition ist unabhängig davon, in welcher Logik man sich befindet, so lange das Konzept der Erfüllbarkeit definiert ist.

a. Geben Sie einen Unsat Core der folgenden Menge prädikatenlogischer Formeln an und begründen Sie ihre Antwort.

$$\{ \quad \forall x\,(p(x) \wedge q(x)), \quad p(a) \vee q(a), \quad \neg p(b) \quad \}$$

(p ist ein einstelliges Prädikatensymbol und a, b sind Konstanten.)

Der (einzige) Unsat Core ist

$$C = \{\forall x\,(p(x) \wedge q(x)), \neg p(b)\}$$

Begründung: C ist ein Unsat Core, weil (1) C unerfüllbar ist (in einem Modell müsste wegen der ersten Formel $p(b)$ wahr und zugleich wegen der zweiten Formel falsch sein, was nicht möglich ist) und zudem die Teilmengen $\{\forall x\,(p(x) \wedge q(x))\}$ und $\{\neg p(b)\}$ erfüllbar sind.

b. Geben Sie ein aussagenlogisches Beispiel an, das zeigt, dass eine unerfüllbare Menge U zwei verschiedene Unsat Cores $C_1 \neq C_2$ haben kann mit $C_1 \cap C_2 \neq \emptyset$.

$$\{ \quad P \wedge Q,\ \neg P,\ \neg Q \quad \}$$

c. Ist entscheidbar, ob eine gegebene endliche Formelmenge C ein Unsat Core ist? Wie hängt das damit zusammen, ob das Erfüllbarkeitsproblem in der jeweiligen Logik entscheidbar ist?

Die Frage, ob C ein Unsat Core ist, ist genau dann entscheidbar, wenn Erfüllbarkeit endlicher Mengen entscheidbar ist. Wenn man Erfüllbarkeit prüfen kann, kann man anhand der Definition prüfen (Teilmengen betrachten), ob C ein Unsat Core ist. Umgekehrt ist eine Menge unerfüllbar genau dann, wenn eine ihrer Teilmengen ein Unsat Core ist. (Für unendliche Mengen ist die Sache komplizierter, und es gibt keinen so einfachen Zusammenhang.)

d. Im Zusammenhang mit automatischem Beweisen ist Erklärbarkeit des Ergebnisses ein wichtiges Konzept. Begründen Sie kurz, warum das Konzept des Unsat Core für die Erklärbarkeit der Ergebnisse, die ein Beweissystem liefert, wichtig ist.

Ein Unsat Core ist wichtig für die Erklärung, warum eine Menge unerfüllbar ist. Denn der Unsat Core ist der Teil einer unerfüllbaren Menge, der widersprüchlich ist. Dies ist insbesondere dann wichtig, wenn der Unsat Core deutlich kleiner als die Gesamtmenge ist. Außerdem kann man aus dem Unsat Core ableiten, welche Axiom (aus einer größeren Menge von Axiomen) benötigt werden, um eine Formel abzuleiten. Für die Erklärung, warum eine erfüllbare Menge tatsächlich erfüllbar ist, taugt das Konzept Unsat Core dagegen nicht.

3 Entscheidungsverfahren für uninterpretierte Funktionssymbole (5 + 2 = 7 Punkte)

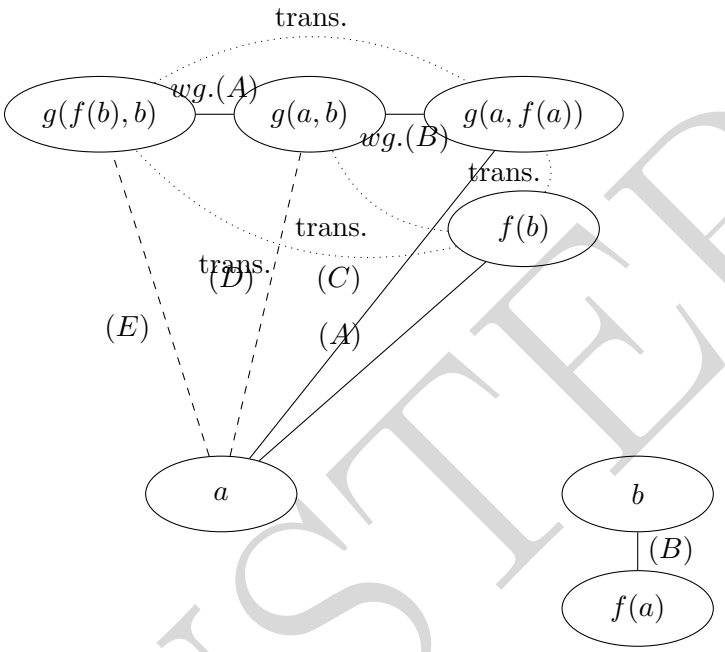
a. Gegeben ist die folgende prädikatenlogische Formel:

$$\underset{(A)}{a \doteq f(b)} \wedge \underset{(B)}{b \doteq f(a)} \wedge \underset{(C)}{a \doteq g(a, f(a))} \wedge \underset{(D)}{\neg a \doteq g(a, b)} \wedge \underset{(E)}{\neg a \doteq g(f(b), b)} \ . \tag{1}$$

Darin sind a, b Konstantensymbole und $f(\cdot), g(\cdot, \cdot)$ sind Funktionssymbole.

Geben Sie den vollständigen Kongruenzgraphen durch die Ausführung des *Shostak*-Algorithmus auf die Formel (1) an.

Durchgezogene Kanten stehen dabei für Gleichheiten und gestrichelte Kanten für Ungleichheiten. Geben Sie an jeder Kante die Begründung an (Name der Gleichheiten).



Bewertungshinweis: Graph sollte vollständig bzgl. der Kanten und Knoten sein modulo den transitiven Kanten.

b. Lesen Sie aus dem fertigen Graphen ab, ob (1) erfüllbar ist oder nicht. Begründen Sie und geben Sie wenn möglich ein erfüllendes Modell an.

Es gibt einen Zyklus im Graphen mit ungerader Anzahl ($n = 1$) an gestrichelten Kante enthält: $g(a, b) = g(a, f(b)) = a \neq g(a, b)$. Daher enthält (1) einen Widerspruch und ist unerfüllbar.

Bewertungshinweis: Erklärungen sollte über den Graphen basieren. Erklärungen ohne solchen Bezug wurden abgestraft.

4 Formalisieren in Prädikatenlogik (PL1)

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

Gegeben sei die prädikatenlogische Signatur $\Sigma = (\{implies\}, \{statement, person, logician, believes\}, \alpha)$. Sie enthält die einstelligen Prädikatensymbole $statement(\cdot)$, $person(\cdot)$ und $logician(\cdot)$, das zweistellige Prädikatensymbol $believes(\cdot, \cdot)$ sowie das zweistellige Funktionssymbol $implies(\cdot, \cdot)$.

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

- das Universum D eine Menge von Aussagen, Personen und Logikerinnen ist,
- das Prädikat $statement(x)$ genau dann wahr ist, wenn x eine Aussage ist,
- das Prädikat $person(x)$ genau dann wahr ist, wenn x eine Person ist,
- das Prädikat $logician(x)$ genau dann wahr ist, wenn x eine Logikerin ist,
- das Prädikat $believes(x, y)$ genau dann wahr ist, wenn Logikerin x glaubt, dass Aussage y wahr ist, und
- die Funktion $implies(x, y)$ die Aussagen x und y auf die Aussage "$x \longrightarrow y$" (y folgt aus x, bzw. unter Voraussetzung x gilt die Folgerung y) abbildet.

Hinweis: Die Aussage "$x \longrightarrow y$" bezeichnen wir forthin als *Implikation*.

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

a. Jede Logikerin ist eine Person.

$$\forall x \, \big(logician(x) \rightarrow person(x)\big)$$

b. Es gibt (mindestens) eine Aussage,

- die alle Logikerinnen glauben und
- die als Folgerung mit beliebiger Voraussetzung dazu führt, dass die Implikation von allen Logikerinnen geglaubt wird.

$$\exists x \, \Big(statement(x) \wedge \forall y \, \big(logician(y) \rightarrow believes(y, x)\big) \\ \wedge \quad \forall y \, \forall z \, \big(logician(y) \wedge statement(z) \rightarrow believes(y, implies(z, x))\big)\Big)$$

c. Wenn eine Logikerin glaubt, dass eine Aussage wahr ist, und sie außerdem glaubt, dass diese erste Aussage eine zweite Aussage impliziert, dann glaubt sie auch diese zweite Aussage.

$$\forall x \, \forall y \, \forall z \, \big(logician(x) \wedge statement(y) \wedge statement(z) \wedge believes(x, y) \wedge believes(x, implies(y, z)) \\ \rightarrow believes(x, z)\big)$$

d. Es gibt (mindestens) eine Folgerung, deren Implikation keine Logikerin glaubt, wenn sie die Voraussetzung glaubt.

$$\exists x \, \Big(statement(x) \wedge \forall y \, \forall z \, \big(logician(y) \wedge statement(z) \wedge believes(y, z) \rightarrow \neg believes(y, implies(z, x))\big)\Big)$$

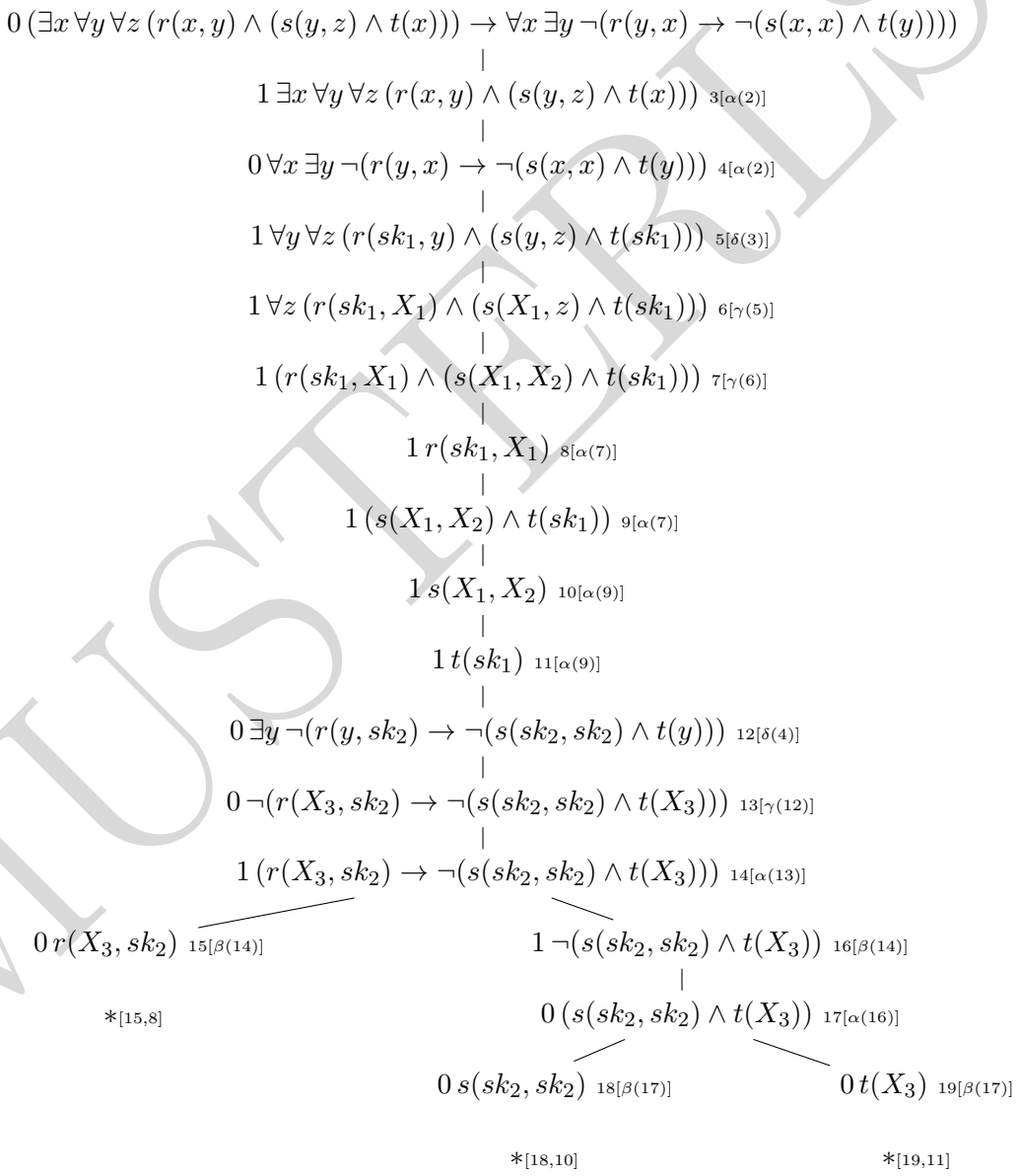
5 Tableaukalkül (10 Punkte)

Beweisen Sie mit Hilfe des Tableaukalküls, dass die folgende prädikatenlogische Formel **allgemeingültig** ist:

$$\Big(\exists x\,\forall y\,\forall z\,\Big(r(x,y)\wedge s(y,z)\wedge t(x)\Big)\Big)\rightarrow\Big(\forall x\,\exists y\,\neg\Big(r(y,x)\rightarrow\neg\Big(s(x,x)\wedge t(y)\Big)\Big)\Big)$$

Notieren Sie dabei:

- bei jeder Erweiterung, durch welche Regelanwendung eine Formel auf dem Tableau entstanden ist,
- bei Abschlüssen die beiden Partner,
- die schließende Substitution.



Die schließende Substitution ist $\{X_1/sk_2, X_2/sk_2, X_3/sk_1\}$

6 Spezifikation mit der Java Modeling Language

(4 + (1,5 + 3,5) = 9 Punkte)

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

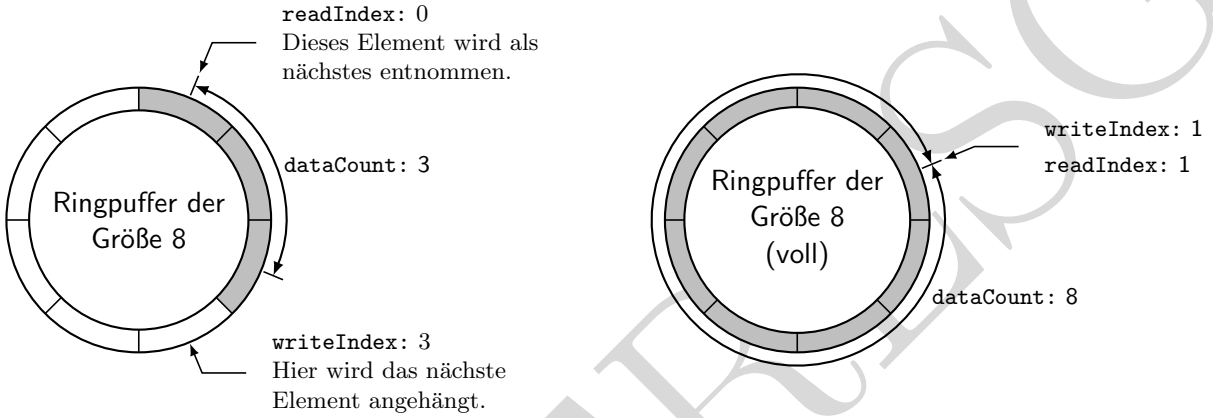
Hinweis: Die Möglichkeit von Integer Overflows können Sie für diese Aufgabe ignorieren.

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

Wenn `m` für ein int-Array aufgerufen wird, das nicht leer ist und nur positive Zahlen enthält, terminiert die Methode, verursacht keine Exception und ändert keine (vor Aufruf der Methode existierenden) Speicherstellen auf dem Heap. Der Rückgabewert der Methode ist dabei ein gemeinsamer Teiler aller Arrayelemente.
Hinweis: Der Teiler ist nicht näher bestimmt, insbesondere würde die Methode den Vertrag auch erfüllen, wenn immer 1 zurückgegeben wird.

Fortsetzung 6 Spezifikation mit der Java Modeling Language

b. Unten sehen Sie einen Teil der Implementierung eines Ringpuffers. Das ist eine Datenstruktur, die eine FIFO-Warteschlange mit beschränkter Kapazität mithilfe eines Arrays umsetzt, das zyklisch gelesen und beschrieben wird. Zum Anhängen an die Warteschlange gibt es eine Methode `put()`, zum Herausnehmen eine Methode `take()`. Hierbei zeigt `writeIndex` immer auf die Stelle im Array, an die als nächstes geschrieben wird, und `readIndex` auf das Element, das als nächstes gelesen (und entfernt) wird. Die Variable `dataCount` zählt die aktuell im Puffer befindlichen Elemente. Zwei mögliche Zustände eines solchen Ringpuffers sehen Sie hier (beispielhaft für die Größe 8):



i. Damit die Datenstruktur korrekt funktioniert, müssen die Indizes (`readIndex`, `writeIndex`) sowie `dataCount` innerhalb bestimmter Intervalle liegen. Formulieren Sie diese Einschränkungen als Objektinvarianten.

Hinweis: Es ist nicht notwendig, die drei Variablen untereinander in Beziehung zu setzen.

ii. Im Fall, dass der Puffer nicht leer ist, soll die Methode `take()` das älteste Element in der Queue zurückgeben, `dataCount` um eins verringern und außerdem `readIndex` „zyklisch“ weiterbewegen. Vervollständigen Sie den Methodenvertrag mit diesen Nachbedingungen, und geben Sie außerdem eine geeignete `assignable`-Klausel an!

```
class RingBuffer {
  final int[] a;    // data
  int readIndex;    // next index to read from
  int writeIndex;   // next index to write to
  int dataCount;    // needed to distinguish between full and empty ...

  //@ invariant 0 <= dataCount  && dataCount <= a.length;
  //@ invariant 0 <= readIndex  && readIndex  < a.length;
  //@ invariant 0 <= writeIndex && writeIndex < a.length;
  /*@ normal_behavior
    @  requires 0 < dataCount;
    @  ensures readIndex == (\old(readIndex) + 1) % a.length;
    @  ensures dataCount == \old(dataCount) - 1;
    @  ensures \result == a[\old(readIndex)];
    @  assignable dataCount, readIndex;
    @*/
  int take() { ... }

  void put(int val) { ... }
}
```

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

Im Folgenden soll mithilfe von LTL das Verhalten eines Geräts mit zwei Lampen beschrieben werden. Eine Lampe ist rot, die andere grün. Dazu wird die Signatur $\Sigma = \{R, G\}$ verwendet:

R ist wahr gdw. die rote Lampe leuchtet

G ist wahr gdw. die grüne Lampe leuchtet

a. Übersetzen Sie folgende LTL-Formeln in natürliche Sprache:

i. $(\Diamond R) \rightarrow (\Box G)$

Wenn irgendwann die rote Lampe leuchtet, dann leuchtet die grüne Lampe immer.

ii. $(R \,\mathbf{U}\, G) \wedge \Box R$

Die rote Lampe leuchtet immer und die grüne Lampe leuchtet irgendwann.

b. Formalisieren Sie die folgenden Aussagen in LTL.

i. Die rote Lampe leuchtet in genau einem Zustand.

Lsg: $\Diamond R \wedge \Box(R \rightarrow \mathbf{X}\Box\neg R)$

ii. Immer wenn die grüne Lampe an ist, ist sie im nächsten Zustand aus und im übernächsten wieder an.

Lsg: $\Box(G \rightarrow \mathbf{X}\neg G \wedge \mathbf{XX}G)$

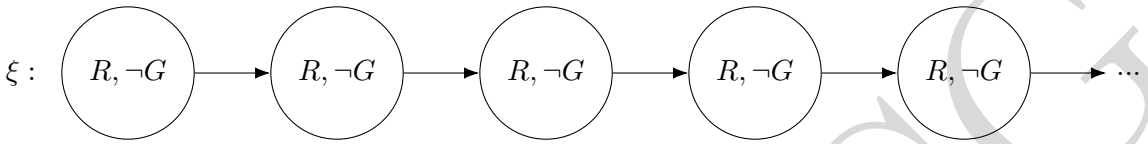
Fortsetzung 7 Lineare Temporale Logik

In der folgenden Teilaufgabe sind jeweils eine LTL-Formel F sowie eine Omega-Struktur ξ gegeben. Die Belegung der Omega-Struktur ändert sich nach dem 5. Zustand nicht mehr.

Geben Sie jeweils an, ob $\xi \models F$ gilt, und begründen Sie kurz.

c. **i.**

$$F \;=\; R \,\mathbf{U}\, G$$

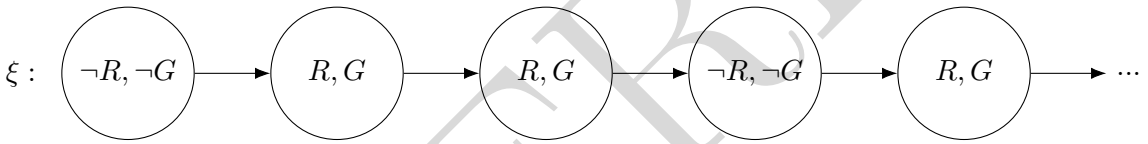


Gilt $\xi \models F$? Begründen Sie kurz.

Nein. *Until* erfordert, dass die grüne Lampe irgendwann leuchtet.

ii.

$$F \;=\; \Box(R \rightarrow \mathbf{XXX}R)$$

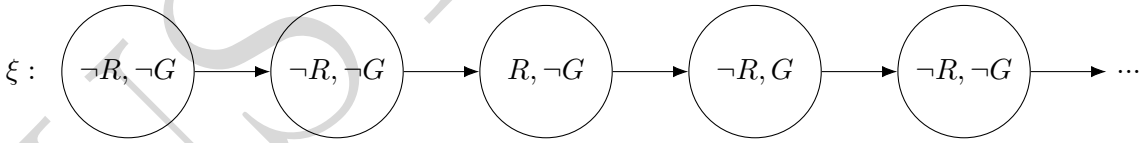


Gilt $\xi \models F$? Begründen Sie kurz.

Ja, denn immer, wenn die rote Lampe leuchtet, leuchtet sie auch drei Zustände später.

iii.

$$F \;=\; \Diamond R \wedge \Diamond G$$



Gilt $\xi \models F$? Begründen Sie kurz.

Ja, denn beide Lampen leuchten irgendwann.