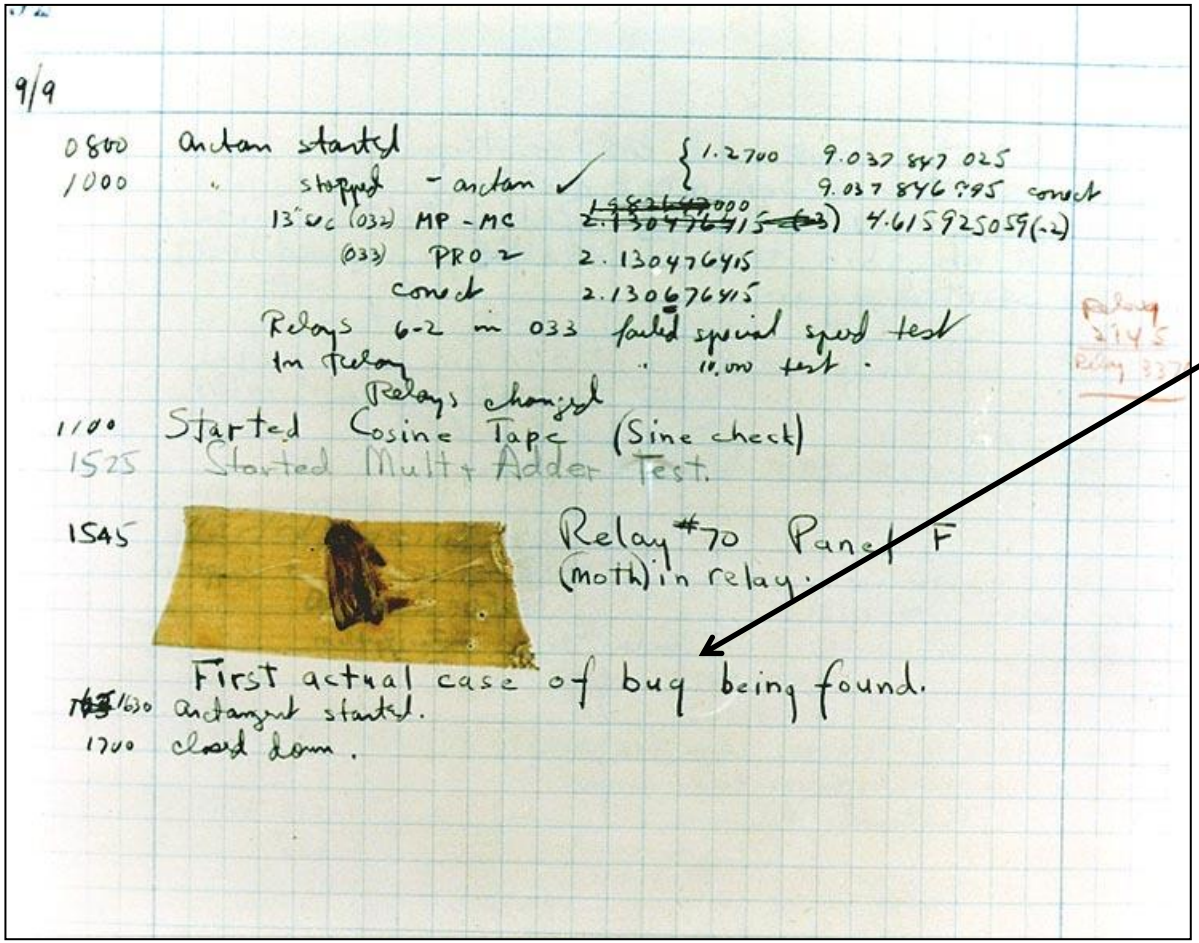
Formale Grundlagen der Informatik 3

Einführung in Model Checking: Spezifikation und Modell, Wiederholung

Prof. Stefan Katzenbeisser
Security Engineering Group
Technische Universität Darmstadt

skatzenbeisser@acm.org
http://www.seceng.informatik.tu-darmstadt.de

Motivation
Softwarefehler: Bugs



„First actual case of bug being found"

Logbuch der Mark II Aiken, 1947

Motivation
Korrektheit von Hardware und Software (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

„Pentium Bug" (1994):

$$4195835 - \frac{4195835}{3145727} * 3145727 = 256$$

- Kosten: 500 Mio. $

Ariane 5 (1996):

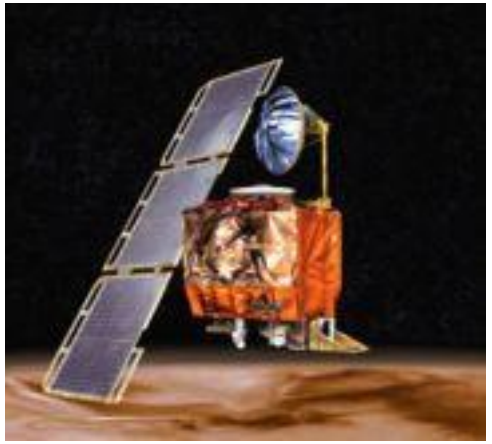
- Softwarefehler in der Verarbeitung von Sensordaten
- Überlauf führte zu Crash eines Teilprogramms
- Falsche Daten für Steuerungscomputer
- Kosten 500 Mio. $ bis 2 Mrd. $ (geschätzt)

SECENG SECURITY ENGINEERING

Motivation
Korrektheit von Hardware und Software (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Mars Climate Orbiter (1999):



- Verwechslung metrischer und britischer Einheiten
- Kosten: 125 Mio. $

LA Air Traffic Control (2004):

- Kontakt zu 400 Flugzeugen verloren
- Crash des Kommunikationssystems durch Unterlauf eines Zählers
- Backup-System versagte

Therac 25 Strahlentherapiegerät (1985):

- Race condition
- >3 Todesfälle

SECENG SECURITY ENGINEERING

Fehlerfreie Software?

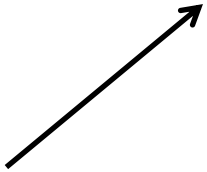
TECHNISCHE UNIVERSITÄT DARMSTADT

Testen:

- Test, ob SW/HW auf bestimmten (kritischen) Eingabedaten korrekt funktioniert
- Testen ist immer unvollständig!
- Problem: repräsentative Testfälle?
- Keine exakten Aussagen über Korrektheit erzielbar

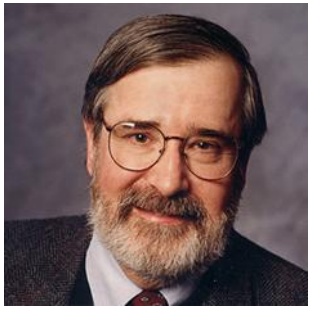
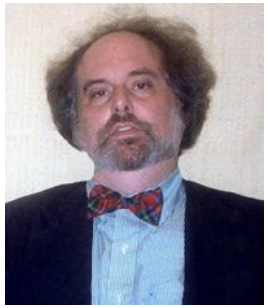
Verifikation:

- Nachweis, dass SW, HW oder Protokolle eine bestimmte Spezifikation erfüllen
- „Beweis" der Korrektheit
- Problem: realistisches Erstellen der Spezifikation
- Recht hohe Komplexität

Fokus dieser Vorlesung 

SECENG SECURITY ENGINEERING

Model Checking: Turing Award 2007

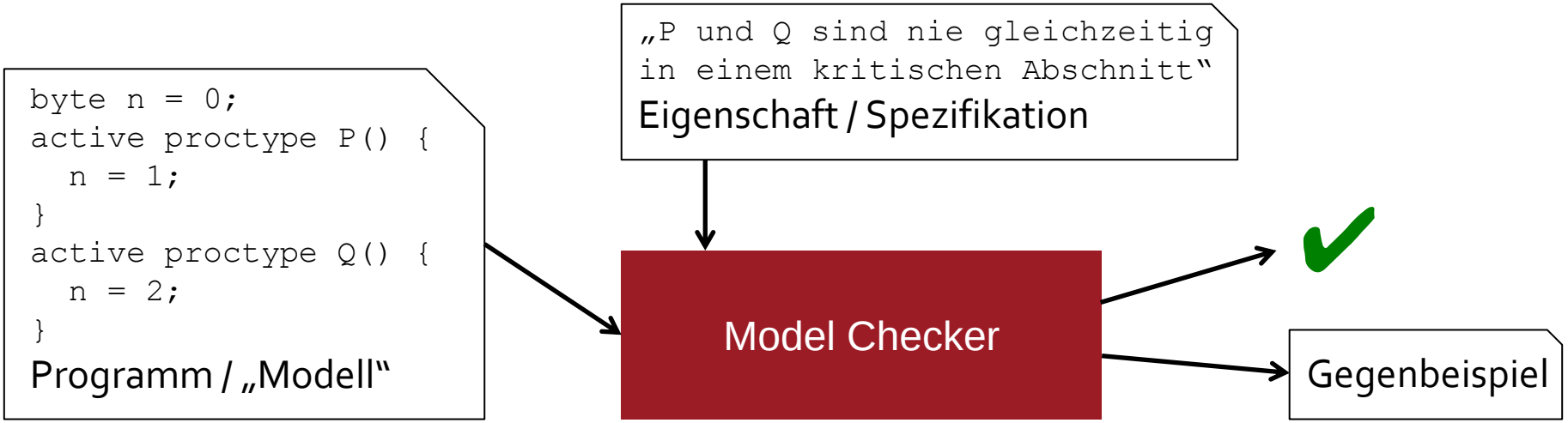

E. Clarke, A. Emerson, J. Sifakis

- Verfahren zum Nachweis der Korrektheit eines Systems
- Testet ein System (**Modell**) gegen eine **Spezifikation**
- Liefert „OK" oder ein **Gegenbeispiel**
- Anwendbar auf Hardware und Software

Model Checking: Turing Award 2007

TECHNISCHE UNIVERSITÄT DARMSTADT

E. Clarke, A. Emerson, J. Sifakis



Formale Methoden

TECHNISCHE UNIVERSITÄT DARMSTADT

Warum?

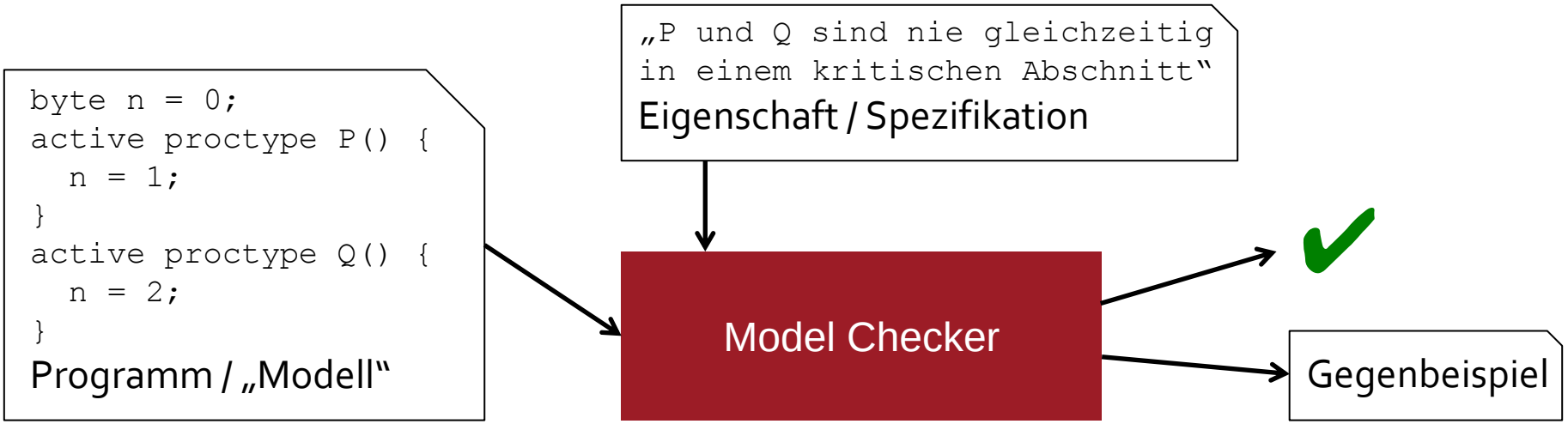
- Exakte (mathematische) Verfahren im Design von Systemen
- Logik, Automaten, …
- Erhöht das Vertrauen in die Korrektheit eines Systems

Wozu?

- Komplementiert andere Analyseverfahren und Tests
- Reduziert Zahl der Bugs
- Reduziert die Entwicklungszeit
- Automatisierbar!

SECENG SECURITY ENGINEERING

Model Checking
Modell und Spezifikation



Modell:

- Abstrakte Repräsentation des Systems
- Meist formalisiert durch einen endlichen Automaten

Spezifikation:

- Eigenschaft, die das System erfüllen soll
- Meist formalisiert durch eine logische Formel

Endliche Automaten, Wiederholung (1)

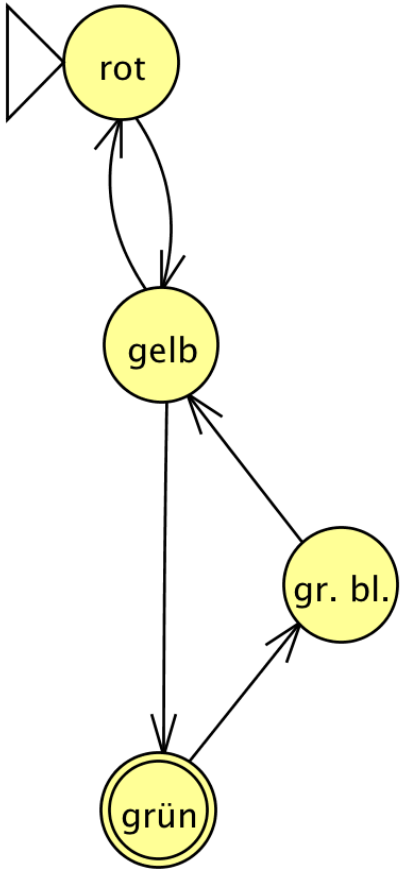
TECHNISCHE UNIVERSITÄT DARMSTADT

Endliche Automaten bestehen aus:

- Zuständen (endlich viele)
- Zustandsübergängen
- einem Startzustand
- beliebig vielen Endzuständen

Beispiel: Ampelschaltung in Österreich

- 4 Zustände
- Zustandsübergänge
- Ein Startzustand
- Endzustand?



SECENG SECURITY ENGINEERING

Endliche Automaten, Wiederholung (2)

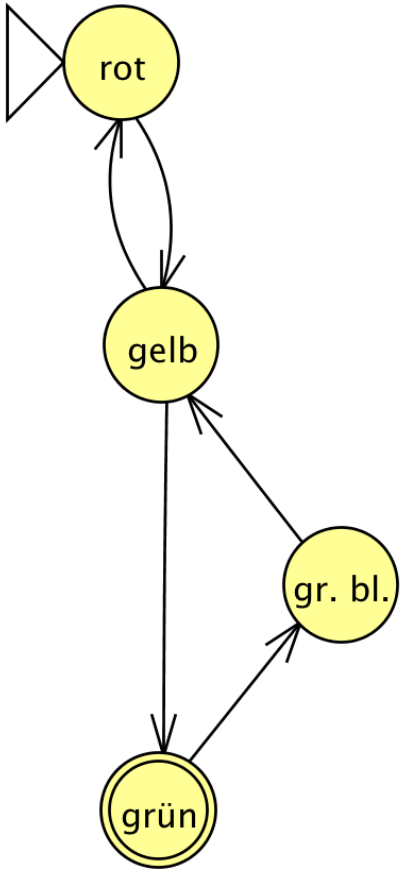
TECHNISCHE UNIVERSITÄT DARMSTADT

Formal:

- Endliche Menge an Zuständen M
- Endliches Alphabet Σ
- Zustandsübergänge: Relation $R \subseteq M \times \Sigma \times M$
- Startzustand $q \in M$
- Menge von Endzuständen: $E \subseteq M$

Endlicher Automat ist definiert als Tupel
$\mathcal{M} = (M, R, q, E, \Sigma)$

Beispiel der Ampelschaltung ➔ Tafel



SECENG SECURITY ENGINEERING

Endliche Automaten, Wiederholung (3)

TECHNISCHE UNIVERSITÄT DARMSTADT

Endlicher Automat:
$\mathcal{M} = (M, R, q, E, \Sigma)$

Wörter sind Elemente aus Σ^* (Kleene-Stern)

Beispiel: $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Menge Σ^* ist i.d.R. **unendlich groß**, die Wörter selbst aber immer **endlich lang**

Ein endlicher Automat **akzeptiert** ein Wort $w_0 w_1 w_2 \ldots w_n$ mit $w_i \in \Sigma$ wenn:

- es eine Sequenz von Zuständen $q_0, q_1, q_2, \ldots, q_{n+1}$ gibt sodass
- der erste Zustand der Startzustand des Automaten ist $(q_0 = q)$
- der letzte Zustand ein Endzustand des Automaten ist $(q_{n+1} \in E)$ und
- ein „gültiger" Zustandsübergang erfolgt: $(q_i, w_i, q_{i+1}) \in R$ für $0 \leq i \leq n$

SECENG SECURITY ENGINEERING

Endliche Automaten, Wiederholung (4)

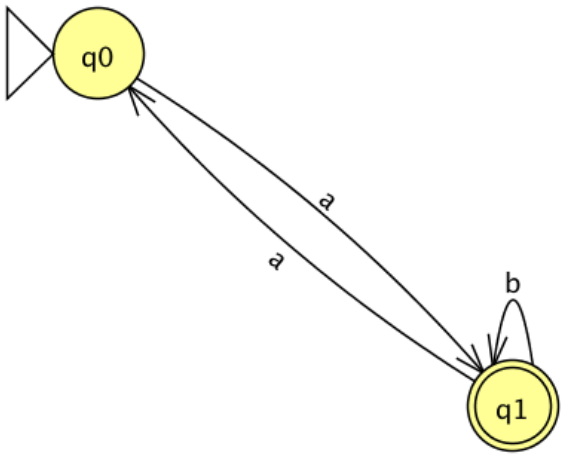
TECHNISCHE UNIVERSITÄT DARMSTADT

Ein endlicher Automat **akzeptiert** ein Wort $w_0 w_1 w_2 \ldots w_n$ mit $w_i \in \Sigma$ wenn:

- es eine Sequenz von Zuständen $q_0, q_1, q_2, \ldots, q_{n+1}$ gibt sodass
- der erste Zustand der Startzustand des Automaten ist $(q_0 = q)$
- der letzte Zustand ein Endzustand des Automaten ist $(q_{n+1} \in E)$ und
- ein „gültiger" Zustandsübergang erfolgt: $(q_i, w_i, q_{i+1}) \in R$ für $0 \leq i \leq n$

Die Menge aller akzeptierten Wörter eines Automaten $\mathcal{M}$ nennt man Sprache des Automaten $L(\mathcal{M})$

Beispiel:



SECENG SECURITY ENGINEERING

Repräsentation der Modelle: Kripke Strukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

Angelehnt an endliche Automaten, aber …

- Modellierung **reaktiver Systeme**, die keinen „Endzustand" kennen
- Relevant sind primär Zustandsänderungen, nicht Eingaben
- Zustände werden mit **„atomaren" Eigenschaften** versehen

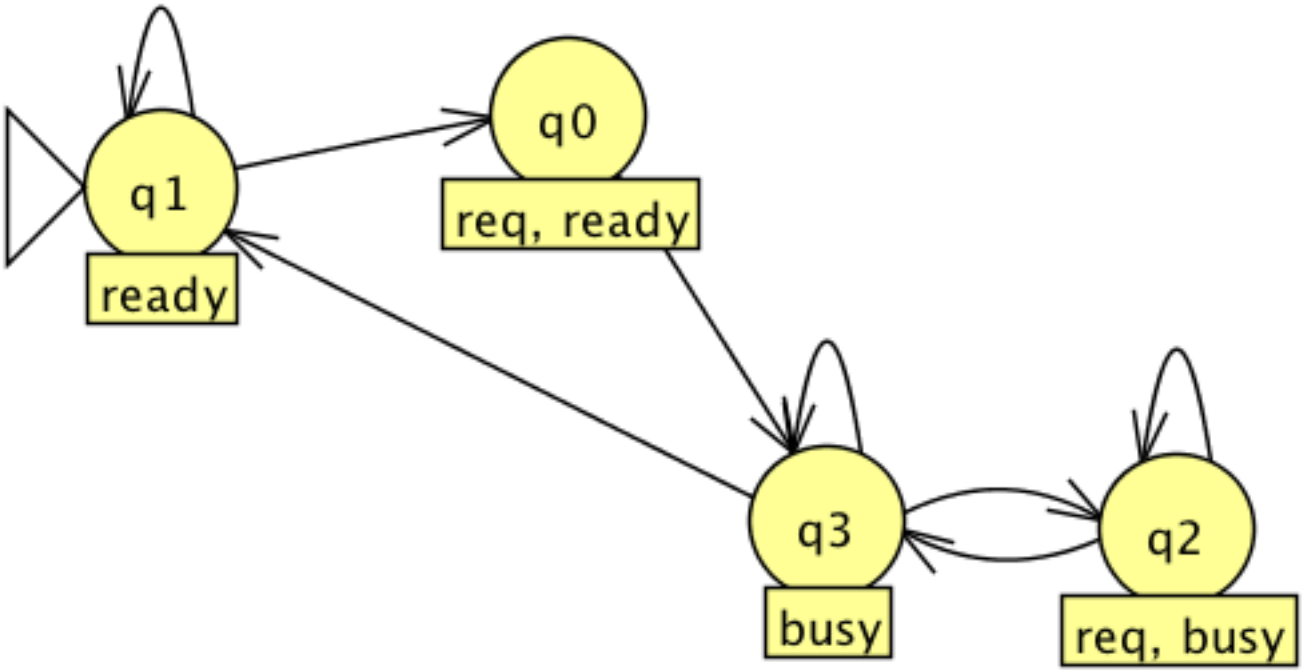
Eine Kripke-Struktur $\mathcal{M} = (S, I, R, L)$ über einer Menge P besteht aus

- einer endlichen Menge an Zuständen S und Anfangszuständen $I \subseteq S$
- einer Übergangsrelation $R \subseteq S \times S$
- und einer Abbildung $L : S \to 2^P$

Eine Kripke-Struktur nennt man **total**, wenn es für jeden Zustand $s \in S$ einen Zustand $s' \in S$ gibt mit $(s, s') \in R$

SECENG SECURITY ENGINEERING

Repräsentation der Modelle: Kripke Strukturen, Beispiel



Repräsentation der Modelle: Kripke Strukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

Eine Kripke-Struktur $\mathcal{M} = (S, I, R, L)$ über einer Menge P besteht aus

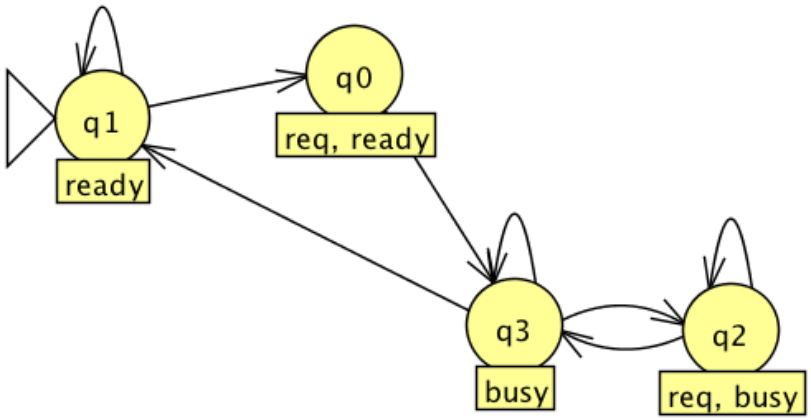
- einer endlichen Menge an Zuständen S und Anfangszuständen $I \subseteq S$
- einer Übergangsrelation $R \subseteq S \times S$
- und einer Abbildung $L : S \to 2^P$

Ein **Pfad** π in einer Kripke-Struktur ist eine **unendliche** Sequenz von Zuständen $\pi = s_0 s_1 s_2 \ldots$ mit $(s, s') \in R$

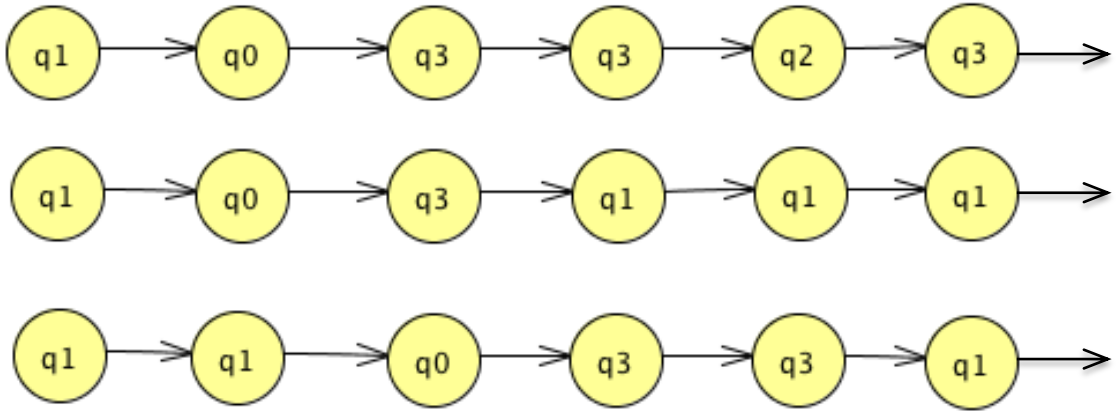
Wir bezeichnen mit π^i den Suffix eines Pfades, der ab der i-ten Position beginnt: $\pi^i = s_i s_{i+1} s_{i+2} \ldots$

SECENG SECURITY ENGINEERING

Repräsentation der Modelle: Kripke Strukturen, Beispiele für Pfade



Einige Pfade:



Repräsentation der Modelle: Kripke Strukturen, Computation Tree

TECHNISCHE UNIVERSITÄT DARMSTADT

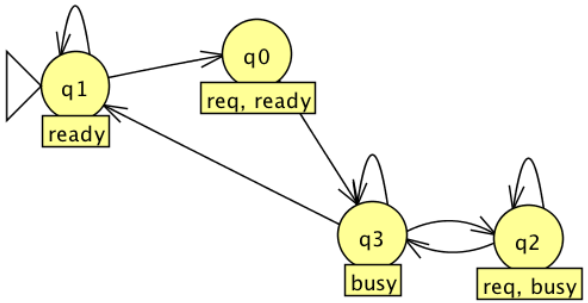
Computation Tree:

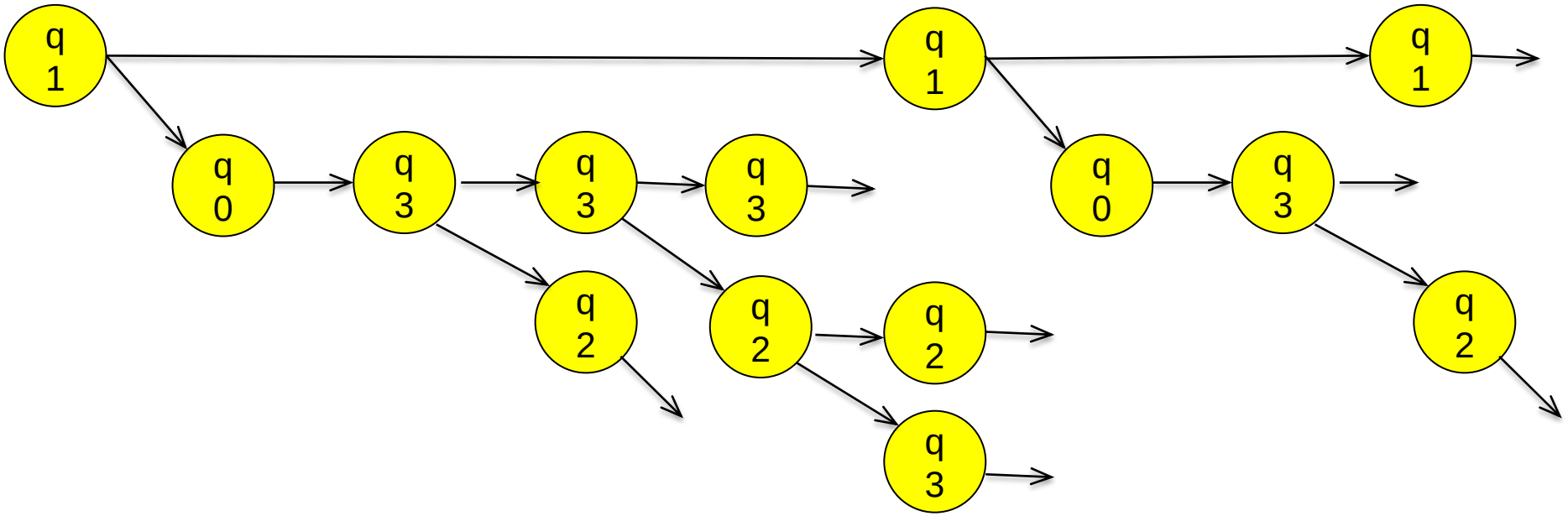
- Zusammenfassung **aller** möglicher Pfade durch eine Kripke-Struktur
- Kompakte Darstellung als Baum
- Zwei Pfade mit gleichem Präfix werden zusammengefasst
- Baum ist immer unendlich tief, da Pfade unendlich lang

- Computation Tree ist „Grundlage" für Verifikation
- Muss implizit gespeichert werden (unendlich groß)

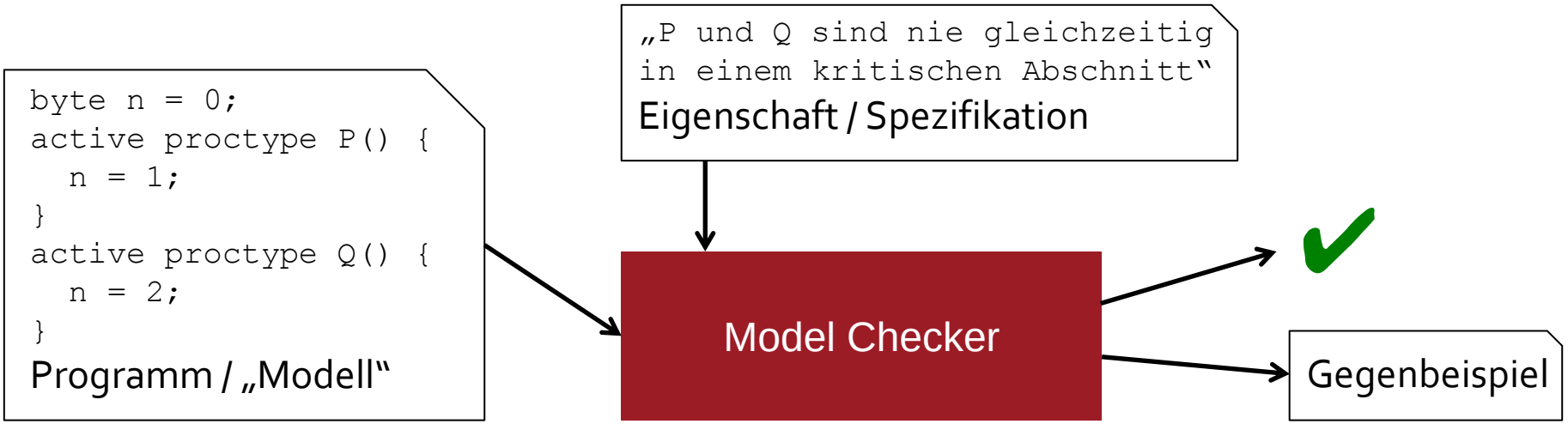
SECENG SECURITY ENGINEERING

Repräsentation der Modelle: Kripke Strukturen, Computation Tree, Beispiel





Model Checking
Modell und Spezifikation



Modell:

- *Kripke-Struktur*
- *meist Spezifikation in „Hochsprache“*

Spezifikation:

?

Repräsentation der Spezifikation: Aussagenlogik, Wiederholung

TECHNISCHE UNIVERSITÄT DARMSTADT

Syntax der Aussagenlogik:

- Menge von Variablen $\mathcal{P}$ (Beispiel: $\mathcal{P} = \{p, q, r, s, \ldots\}$)
- Junktoren: $\vee, \wedge, \neg, \leftrightarrow, \rightarrow$
- Konstanten: $\top, \bot$

Die Menge der aussagenlogischen Formen ist die *kleinste* Menge für die gilt:

- Die Konstanten $\top, \bot$ sowie alle Variablen in $\mathcal{P}$ sind aussagenlogische Formeln.
- Sind φ und ψ aussagenlogische Formeln, so sind auch $\neg\varphi, (\varphi \vee \psi), (\varphi \wedge \psi), (\varphi \leftrightarrow \psi), (\varphi \rightarrow \psi)$ aussagenlogische Formeln. (Induktive Definition)

SECENG SECURITY ENGINEERING

Repräsentation der Spezifikation: Syntax der Aussagenlogik, Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

Die Menge der aussagenlogischen Formen ist die *kleinste* Menge für die gilt:

- Die Konstanten $\top, \bot$ sowie alle Variablen in $\mathcal{P}$ sind aussagenlogische Formeln.
- Sind φ und ψ aussagenlogische Formeln, so sind auch $\neg\varphi, (\varphi \vee \psi), (\varphi \wedge \psi), (\varphi \leftrightarrow \psi), (\varphi \rightarrow \psi)$ aussagenlogische Formeln. (Induktive Definition)

Beispiele: Sind die folgenden Strings aussagenlogische Formeln?

Nehmen wir an dass $\mathcal{P} = \{p, q, r, s, \ldots\}$

$(\top \rightarrow p)$	$(p \rightarrow (q\vee))$	$(\bot \wedge (p \rightarrow (q \wedge r)))$
$(\top \rightarrow))$	$(p \rightarrow (q \vee q))$	$(p \rightarrow (q \rightarrow (r \rightarrow s)))$
$((p(q \vee r)) \wedge p)$	$(\top \rightarrow \bot)$	$(p * (q \rightarrow (r \rightarrow s)))$

SECENG SECURITY ENGINEERING

Repräsentation der Spezifikation: Semantik der Aussagenlogik, Wiederholung (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Interpretation: Abbildung $I : \mathcal{P} \to \{\top, \bot\}$
weist jeder Variable entweder den Wahrheitswert TRUE oder FALSE zu.

Beispiel: $\mathcal{P} = \{p, q\}$

	p	q
I_1	$\bot$	$\bot$
I_2	$\bot$	$\top$
I_3	$\top$	$\bot$
I_4	$\top$	$\top$

Wie werte ich nun eine Formel wie $(p \to (q \to p))$ unter einer Interpretation aus?

SECENG SECURITY ENGINEERING

Repräsentation der Spezifikation: Semantik der Aussagenlogik, Wiederholung (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Auswertungsfunktion:
Erweiterung der Abbildung auf aussagenlogische Formeln über eine Auswertungsfunktion v_I, die einer Formel einen Wahrheitswert zuweist

$$v_I(p) = I(p) \quad v_I(\top) = \top \quad v_I(\bot) = \bot$$

$$v_I(\neg\varphi) = \begin{cases} \top & v_I(\varphi) = \bot \\ \bot & \text{sonst} \end{cases} \qquad v_I(\varphi \leftrightarrow \psi) = \begin{cases} \top & v_I(\varphi) = v_I(\psi) \\ \bot & \text{sonst} \end{cases}$$

$$v_I(\varphi \wedge \psi) = \begin{cases} \top & v_I(\varphi) = \top \text{ und } v_I(\psi) = \top \\ \bot & \text{sonst} \end{cases}$$

$$v_I(\varphi \vee \psi) = \begin{cases} \top & v_I(\varphi) = \top \text{ oder } v_I(\psi) = \top \\ \bot & \text{sonst} \end{cases}$$

$$v_I(\varphi \rightarrow \psi) = \begin{cases} \top & v_I(\varphi) = \bot \text{ oder } v_I(\psi) = \top \\ \bot & \text{sonst} \end{cases}$$

SECENG SECURITY ENGINEERING

Repräsentation der Spezifikation: Semantik der Aussagenlogik, Wiederholung (3)

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel: $\mathcal{P} = \{p, q\}$

	p	q	$(p \rightarrow (q \rightarrow p))$
I_1	$\bot$	$\bot$	
I_2	$\bot$	$\top$	
I_3	$\top$	$\bot$	
I_4	$\top$	$\top$	

Wir bestimmen: $v_{I_2}(p \rightarrow (q \rightarrow p))$

➔ Tafel

SECENG SECURITY ENGINEERING

Repräsentation der Spezifikation: Erfüllbarkeit, Gültigkeit (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Erfüllbarkeit:

- Falls $v_I(\varphi) = \top$ schreiben wir auch $I \models \varphi$
  „ I erfüllt φ “
- Eine aussagenlogische Formel φ ist **erfüllbar**, wenn es eine Interpretation I gibt mit $I \models \varphi$

Gültigkeit:

- Eine Formel φ ist **gültig**, wenn für alle Interpretationen I gilt: $I \models \varphi$
- Wir schreiben dann: $\models \varphi$

SECENG SECURITY ENGINEERING

Repräsentation der Spezifikation: Erfüllbarkeit, Gültigkeit (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine aussagenlogische Formel φ ist **erfüllbar**, wenn es eine Interpretation I gibt mit $I \models \varphi$
- Eine Formel φ ist **gültig**, wenn für **alle** Interpretationen I gilt: $I \models \varphi$

Beispiele: Sind die folgenden aussagenlogische Formeln erfüllbar und/oder gültig?

p

$p \vee \neg p$

$(p \vee \neg p) \rightarrow r$

$(p \wedge (\neg p \vee q)$

$(p \rightarrow (q \rightarrow p))$

$(p \wedge \neg p) \rightarrow r$

SECENG SECURITY ENGINEERING

Repräsentation der Modelle: Modellierung durch Logik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Grundlegende Eigenschaften von Programmen sind in Aussagenlogik formulierbar
- Erste Idee: beschreibe alle mögliche Zustände eines Programms P als aussagenlogische Formel φ_P ➔ Meist zu komplex!

- **Formalisierung bestimmter wichtiger Aussagen**
  - *Der Drucker ist gerade beschäftigt*
  - *Der Drucker ist nicht beschäftigt und nimmt Aufträge an*
  - *Es existieren keine Aufträge für den Drucker*
  - *Der Inhalt des Registers R1 ist der Wert 6*
  - *…*

SECENG SECURITY ENGINEERING

Repräsentation der Modelle: Modellierung durch Logik, Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

Formalisierung bestimmter Aussagen:

- *Der Drucker ist gerade beschäftigt*

  $P = \{idle\}, \varphi_1 = \neg idle$

- *Der Drucker ist nicht beschäftigt und nimmt Aufträge an*

  $P = \{idle, accept\_req\}, \varphi_2 = idle \wedge accept\_req$

- *Es existieren keine Aufträge für den Drucker*

  $P = \{N_0, N_1, N_2, \ldots, N_7\}, \varphi_3 = \neg N_0 \wedge \neg N_1 \wedge \ldots \wedge \neg N_7$

SECENG SECURITY ENGINEERING

Repräsentation der Modelle: Grenzen der Aussagenlogik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Modellierung von „Zeit“ ist schwierig

- Modellierung der folgenden Aussagen nicht möglich:
  - Der Inhalt des Registers R1 wird irgendwann im Ablauf des Programms den Wert 0 enthalten.
  - Der Inhalt des Registers R1 ändert sich unendlich oft.
  - Jeder Request wird irgendwann auch beantwortet.
  - Das Programm wird immer wieder in einen initialen Zustand gelangen.
  - …

- Hierfür ist eine Logik nötig die zeitliche Abhängigkeiten berücksichtigt!
  ➔ nächste Woche!

SECENG SECURITY ENGINEERING