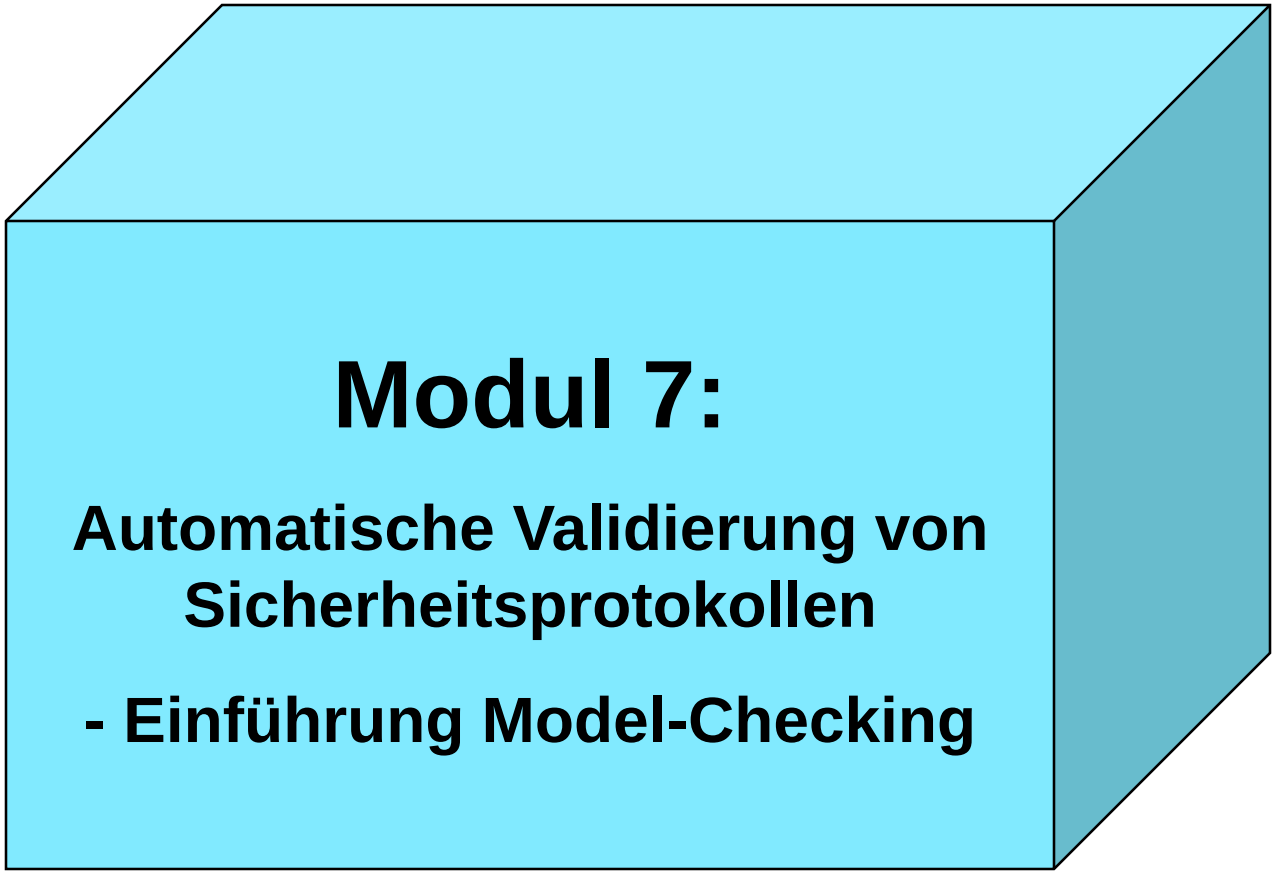
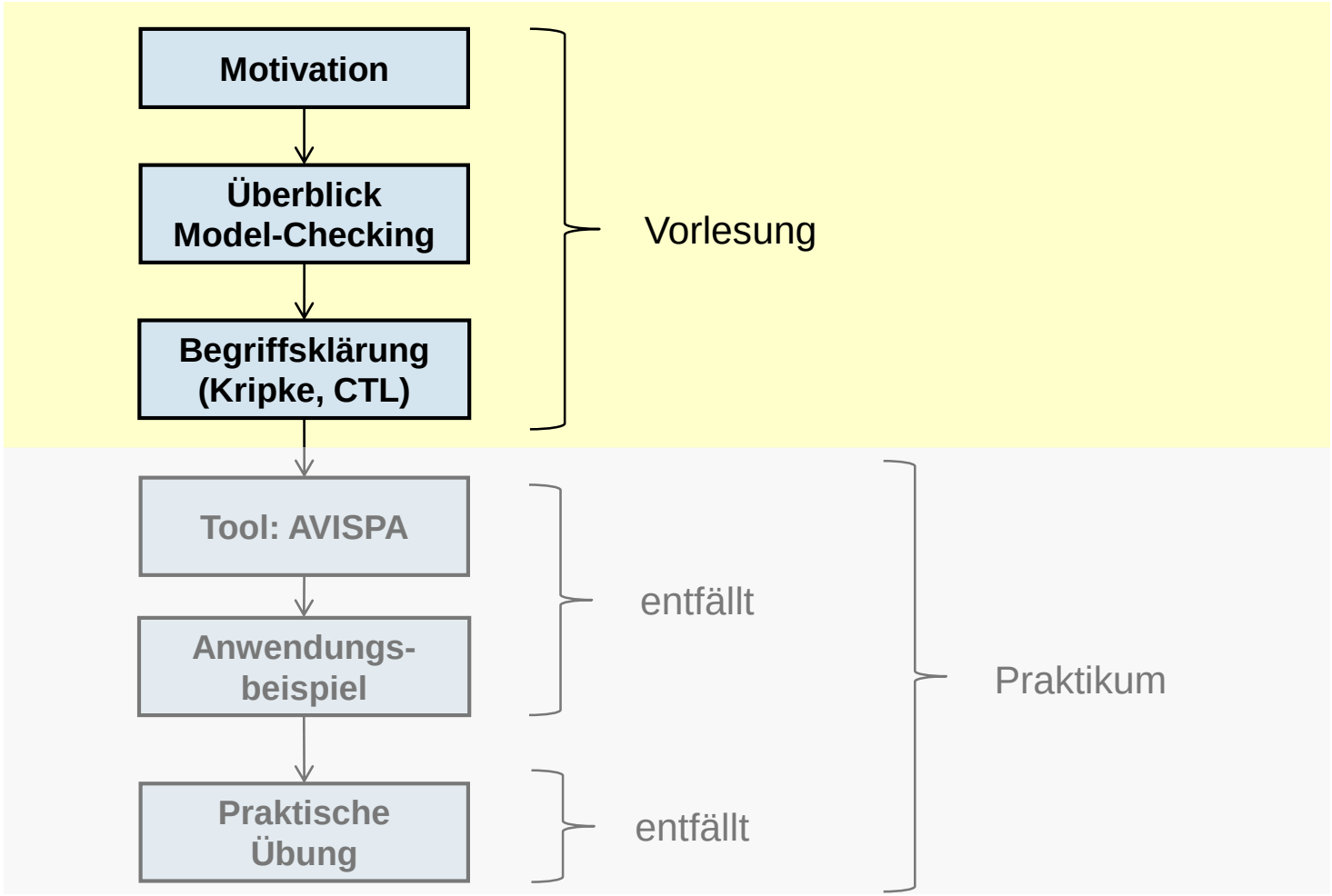

Modul 7:
Automatische Validierung von Sicherheitsprotokollen
- Einführung Model-Checking

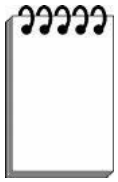
Automatische Validierung von Protokollen - Lehrkonzept



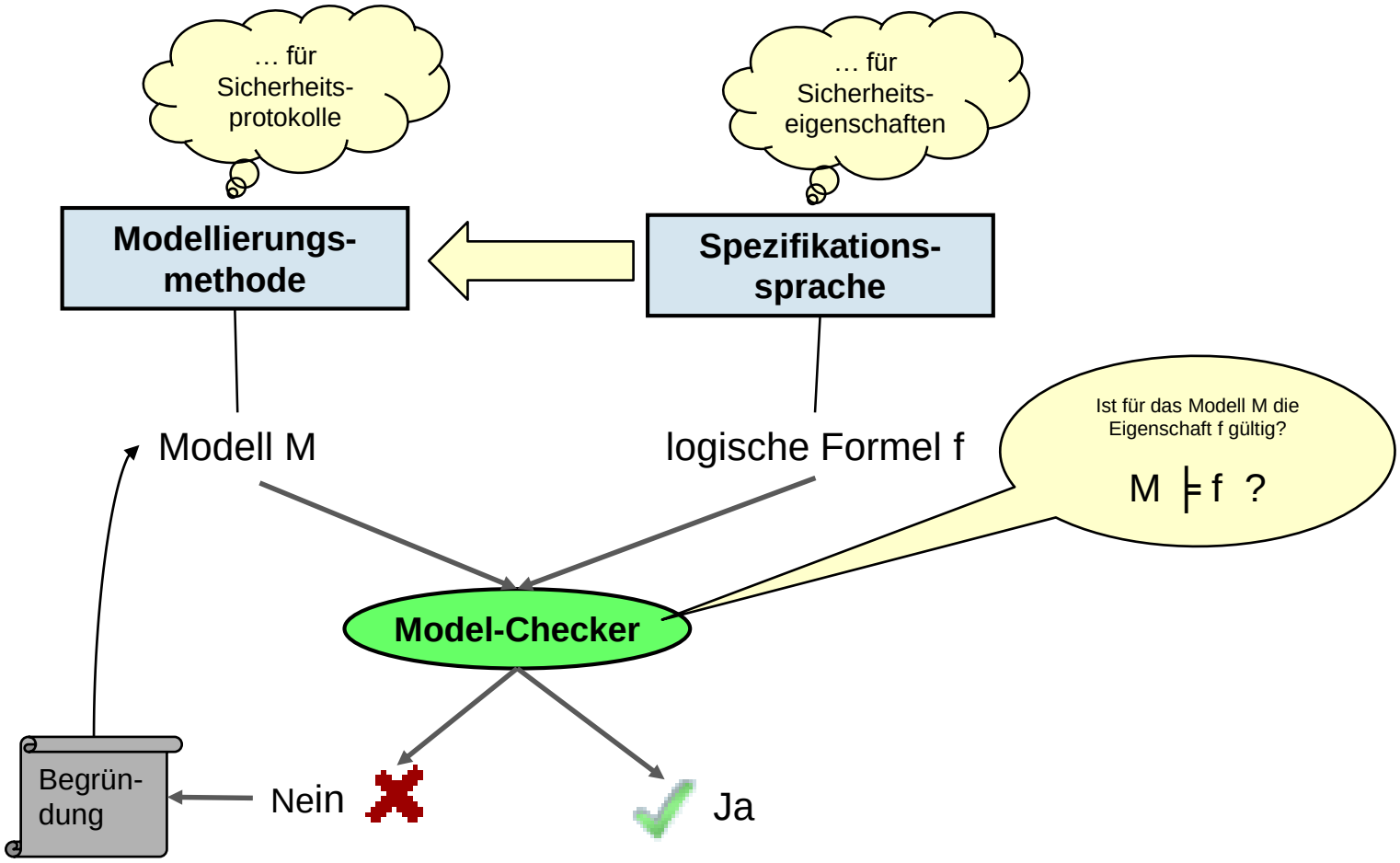
Motivation

Sicherheit von Kommunikationsprotokollen:

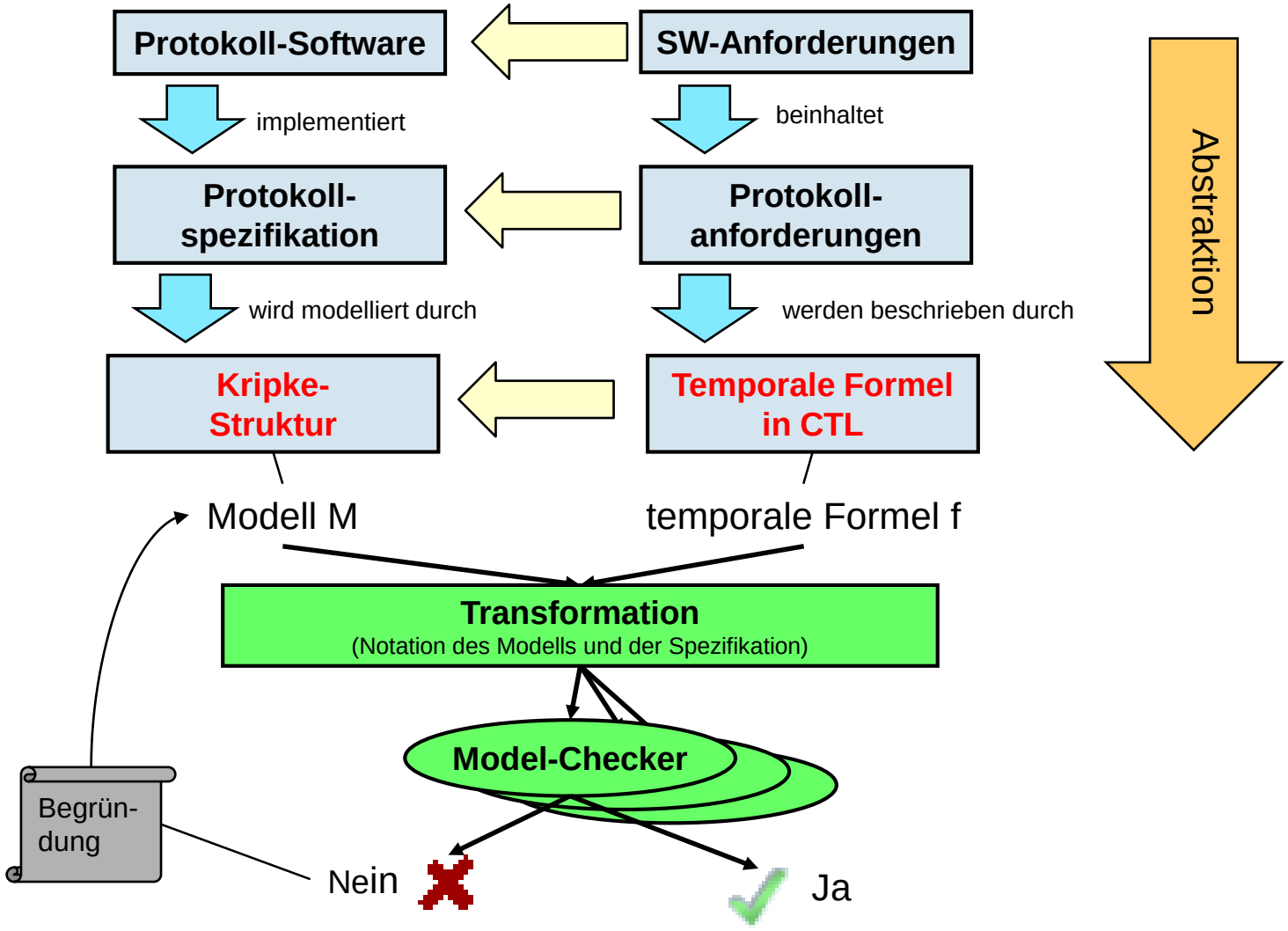
- **Fragestellung**
  Die Frage „Ist das Protokoll sicher?“ ist etwas zu allgemein.
- **Methodik**
  Wie „bekomme“ ich das Protokoll sicher?



Übersichtsfolien – Model-Checking allgemein



Übersichtsfolien - Automatische Validierung von Protokollen

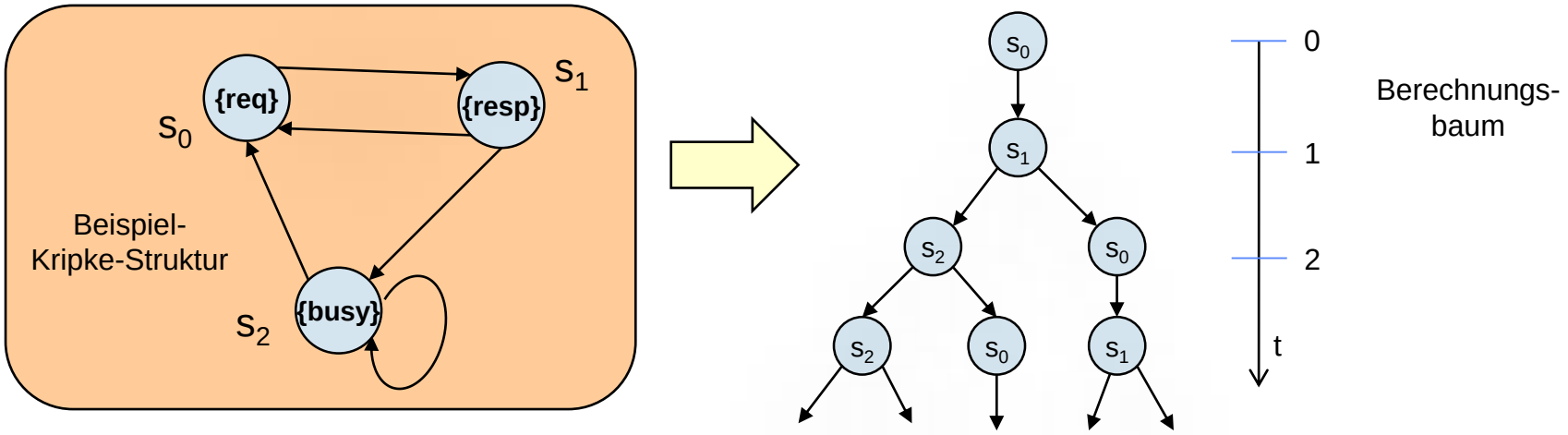


Definition Kripke-Struktur

Sei $P = \{p_1, \ldots, p_n\}$ eine endliche Menge aussagenlogischer Atome.

Eine Kripke-Struktur M über P ist ein Quadrupel $M = (S, S_0, \rightarrow, \mu)$, wobei:

- **S eine endliche Menge von Zuständen,**
- **$S_0 \subseteq S$ eine Menge von Anfangszuständen,**
- **$\rightarrow \subseteq S \times S$ eine totale Transitionsrelation über S und**
- **$\mu : S \rightarrow Pot(P)$ eine Kennzeichnungsfunktion ist, die jedem Zustand $s \in S$ die Menge der als wahr interpretierten Atome $\mu(s) \subseteq P$ zuordnet.**



Beispiel Kripke-Struktur über M = (S, S_0, →, μ) über P

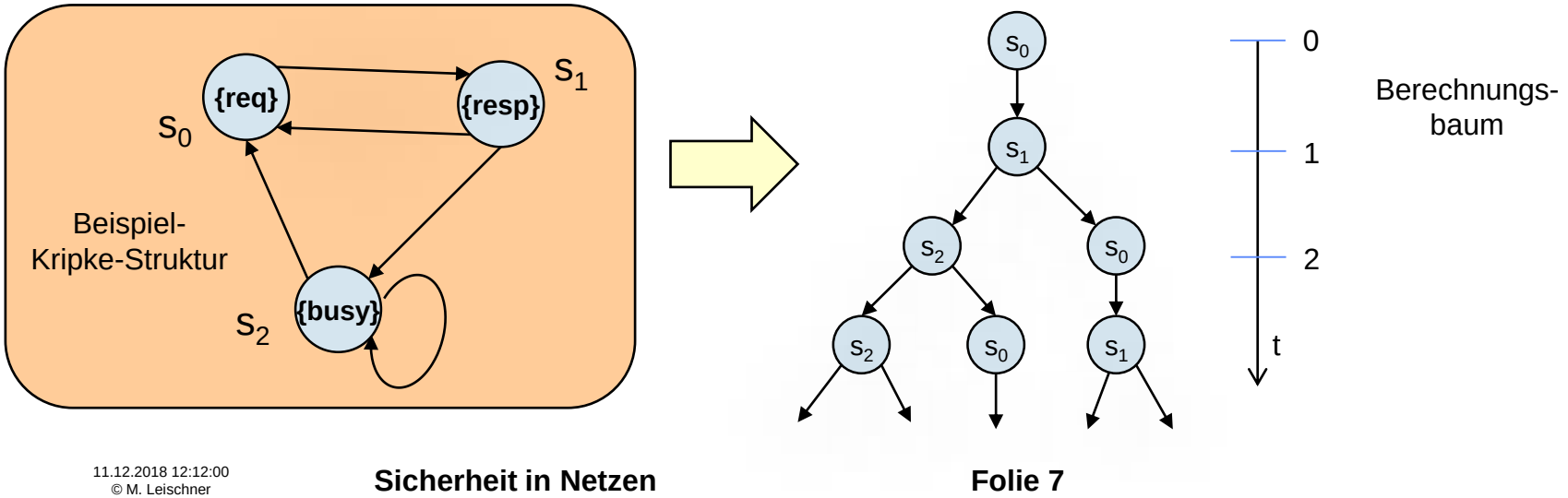
P = {req, resp, busy}

S = {s_0, s_1, s_2}

S_0 = {s_0}

→ = {(s_0, s_1), {(s_1, s_0), {(s_1, s_2), {(s_2, s_0), {(s_2, s_2), } ⊆ S × S

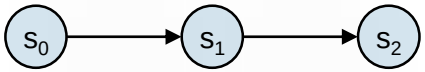
μ : S → Pot(P) mit μ(s_0) ={req}, μ(s_1) ={resp}, μ(s_3) ={busy}



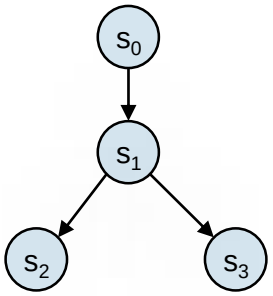
Bemerkungen zur Kripke-Struktur

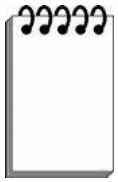
- **Unterschied/Gemeinsamkeiten:**
  - **endlicher Automat**
  - **Kripke-Struktur**

- **Mit einer Kripke-Struktur lässt sich ein nicht-deterministisches Verhalten beschreiben.**
- **Um Eigenschaften von Kripke-Systemen zu beschreiben, muss sowohl der zeitliche Aspekt als auch der Ausführungsaspekt zusätzlich berücksichtigt werden.**
- **Spezifikation des zeitlichen Aspekts:**
  LTL (Linear temporal logic)



- **Spezifikation des zeitlichen und des Ablaufaspekts:**
  CTL (Computation tree logic)





Beschreibungselement der CTL

- **Aussagenlogische Operatoren:**
  true, false, ∧, ∨, ¬ und die davon abgeleiteten Operatoren wie ⇒

- **Temporale Operatoren („Wann gilt etwas?“):**
  - **X ϕ** ϕ gilt im nächsten Zeitpunkt bzw. Zustand („nexttime“).
  - **F ϕ** ϕ gilt irgendwann jetzt oder später („eventually“, „sometime“, „future“).
  - **G ϕ** ϕ gilt zu jedem Zeitpunkt bzw. Zustand („immer“, „always“, „globally“).
  - **ϕ until ψ** es gilt ϕ bis ψ gilt und
    ψ gilt auch irgendwann einmal in der Zukunft („starker Operator“).

- **Pfadquantoren („Bei welchen Abläufen gilt etwas“?)**
  - **E ϕ** es gibt einen Pfad, der ϕ erfüllt.
  - **A ϕ** alle Pfade erfüllen ϕ .

Formale Definition der CTL (Syntax)

Sei P eine Menge von atomaren Aussagen. Dann gilt:

- **Jede atomare Aussage ist eine CTL-Formel**
- **Wenn ϕ und ψ CTL-Formeln sind, dann auch:**
  true, false, ¬ ϕ, ϕ ∧ ψ , AXϕ, EXϕ, AFϕ, EFϕ, AGϕ, EGϕ, A(ϕ until ψ),
  E(ϕ until ψ)

Bemerkung:

- **Das Ganze lässt sich noch kompakter (aber unübersichtlicher) formulieren.**
- **<u>Prinzip</u>: jeder temporallogischen Formel wird ein Pfadquantor vorgestellt.**
  D.h. es wird gesagt, ob die temporallogische Formel auf jedem Pfad gilt oder, ob nur einen Pfad existiert, so dass sie gilt.
- **Bis jetzt haben wir rein syntaktisch definiert, was korrekte CTL-Formeln sein sollen. Die Bedeutung existiert bislang nur informell im „Hinterkopf“**
- **Im folgenden soll nun exakt die Semantik der Formeln definiert werden.**
  → Dazu benötigen wir die Kripke-Struktur.

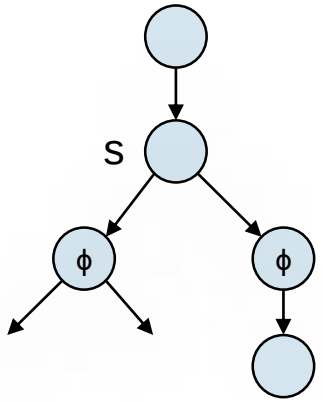
Formale Definition der CTL (Semantik)

Sei M = (S, S_0, →, μ) eine Kripke-Struktur über einer endlichen Menge P aussagenlogischer Atome.

Im folgenden wird definiert, wann für die Kripke-Struktur M eine Formel ϕ im Zustand s ∈ S gültig ist.

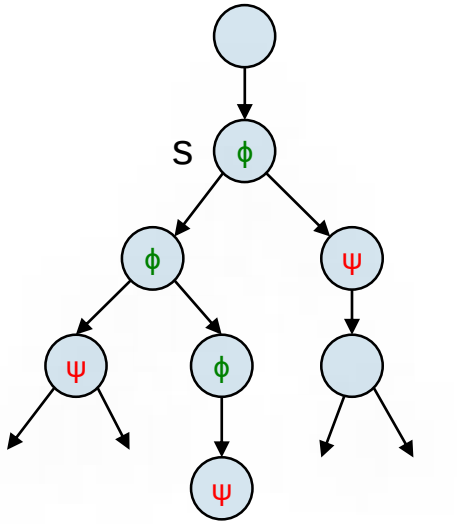
Notation: (M, s) ⊨ ϕ

- **(M, s) ⊨ true** — **immer**
- **(M, s) ⊨ p** — **genau dann, wenn p ∈ μ(s), wobei p ∈P.**
- **(M, s) ⊨ ¬ ϕ** — **genau dann, wenn nicht (M, s) ⊨ ϕ gilt.**
- **(M, s) ⊨ ϕ ∧ ψ** — **genau dann, wenn (M, s) ⊨ ϕ und (M, s) ⊨ ψ gilt.**
- **(M, s) ⊨ AXϕ** — **genau dann, wenn ∀ s' ∈ S mit s → s': (M, s') ⊨ ϕ .**
- **(M, s) ⊨ EXϕ** — **genau dann, wenn ∃ s' ∈ S mit s → s': (M, s') ⊨ ϕ .**



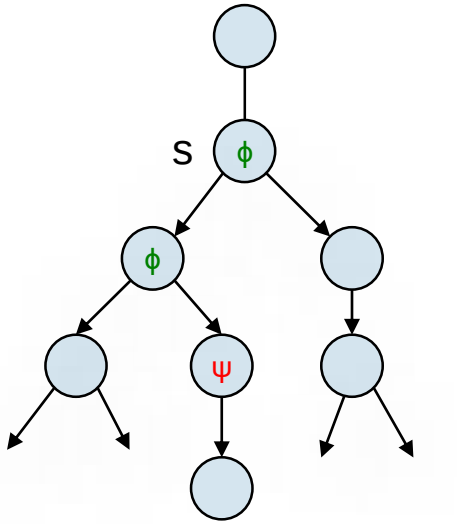
Formale Definition der CTL (Semantik) - Fortsetzung

- **(M, s) ⊨ A(ϕ until ψ)** **genau dann, wenn für jeden Pfad**
  $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow \ldots$ mit $s = s_0$ gilt:
  $\exists\, k \geq 0$, so dass $(M, s_k) \models \psi$ und
  $\forall\, j$ mit $0 \leq j < k$ gilt $(M, s_j) \models \phi$



Formale Definition der CTL (Semantik) - Fortsetzung

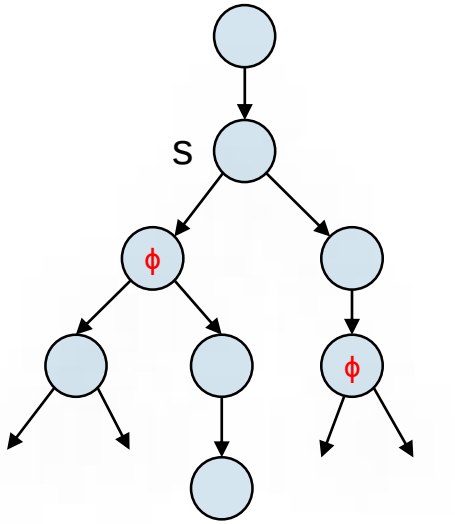
- **$(M, s) \models E(\phi \text{ until } \psi)$** **genau dann, wenn es einen Pfad**
  $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow \ldots$ gibt, so dass:
  $\exists\, k \geq 0$, so dass $(M, s_k) \models \psi$ und
  $\forall\, j$ mit $0 \leq j < k$ gilt $(M, s_j) \models \phi$



Formale Definition der CTL (Semantik) - Fortsetzung

- **(M, s) ⊨ AFϕ** **genau dann, wenn (M, s) ⊨ A(true until ϕ)**

„Auf allen von s ausgehenden Pfaden gilt irgendwann einmal die Formel ϕ“



Formale Definition der CTL (Semantik) - Fortsetzung

- $(M, s) \models EF\ \phi$ genau dann, wenn $(M, s) \models E(\text{true until } \phi)$
  „Es gibt einen von s ausgehenden Pfaden, auf dem irgendwann die Formel ϕ gilt“
- $(M, s) \models AG\phi$ genau dann, wenn $(M, s) \models \neg (EF \neg \phi)$
  „Auf allen von s ausgehenden Pfaden gilt immer die Formel ϕ“
- $(M, s) \models EG\ \phi$ genau dann, wenn $(M, s) \models \neg (AF \neg \phi)$
  „Es gibt einen von s ausgehenden Pfaden, auf dem die Formel ϕ immer gilt“

- Eine CTL-Formel ϕ heißt für eine Kripke-Struktur M genau dann gültig, wenn ϕ in allen Anfangszuständen von M gilt.

Notation: $\models \phi$

Beispiel von Aussagen

Temporallogische Aussagen zur Beispiel-Kripke-Struktur auf Folie 6:

- **Gegeben folgende Formel:**

  req ==> AF resp

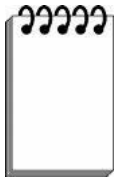
  „Zu jedem Request erfolgt irgendwann einmal ein Response“

  Beachte: Dies gilt definitionsgemäß nur für den Anfangszustand.

- **Verbesserung/Berichtigung der Formel zu:**

  AG(req ==> AF resp)

- **Formulieren Sie in einer temporallogischen Formel folgende Aussagen:**
  - **Das System kann nie gleichzeitig in den Zustand req und resp gelangen.**
  - **Egal in welchem Zustand das System ist, es kann wieder in den Zustand busy gelangen.**
  - **Immer wenn das System in den Zustand busy kommt, bleibt es in diesem Zustand, bis ein req auftritt.**



Explizites Model Checking („On-the-Fly-Model-Checking“):

- Expliziter Aufbau des Berechnungsbaums.
- Alle Zustände und Übergänge des zu modellierenden Systems werden explizit konstruiert.
- Jeder Zustand, jeder Übergang und jede Eigenschaft wird auf die vom Benutzer vorgegebene Spezifikation hin überprüft.
- Speicherung der ausgewerteten Zustände (aufwändig).
- On-the-Fly-Model-Checker setzen verschiedene Suchstrategien ein
  - Tiefensuche (Depth-First Search)
  - Breitensuche (Breadth First Search)
  - Beschränkte Tiefensuche (Depth Limited Search)
- Reduktion des Suchraums kann auch durch Beschränkung auf „relevante“ Teile des Zustandsgraphen erfolgen.

Bewertung: explizites Model-Checking

<u>Problem</u> beim expliziten Model Checking („On-the-Fly-Modell-Checking“):

- **Bei großer Anzahl von Zuständen und beobachtbaren Eigenschaften lässt sich die Aussage über eine Kripke-Struktur nicht mehr effizient verifizieren („State-Explosion-Problem“, „exponentielles Wachstum“).**
- **Nur einfache Systeme sind verifizierbar.**
- **Für praktische Anwendungen in den meisten Fällen unbrauchbar.**

Symbolisches Model-Checking

Lösung: symbolische Darstellung der Kripke Struktur

- **Der Zustandsgraph wird nicht mehr explizit aufgebaut, sondern kompakt durch eine boolesche Formel „symbolisch“ repräsentiert.**
- **Boolsche Formel wird als Ordered Binary Decision Diagram (OBDD) interpretiert.**
- **OBDDs lassen sich vereinfachen (reduced OBDD, ROBDD)**
  - **Symmetrien in Berechnungsbaum werden geschickt zusammengefasst**
  - **kein expliziter Aufbau des Berechnungsbaumes, sondern Reduktion des Zustandsraumes durch Reduktionsregeln**
- **Temporallogische Aussagen über ROBDDs lassen sich in der Zeitkomplexität $O(n^2)$ verifizieren (→ Durchbruch beim Model-Checking)**
- **Model-Checking wird für praktische Anwendungen (z.B. Sicherheitsprotokolle) möglich.**

Symbolische Darstellung der Beispiel-Kripke-Struktur (Folie 6)

Methode:

- **Schritt 1: Kodiere die Zustände als Bitvektoren.**

  Dies geschieht mittels einer injektiven Funktion

  $$\text{code: } S \rightarrow \{0,1\}^n \text{ mit } 2^n \geq \text{card}(S)$$

- **Schritt 2: Definiere Transition als boolesche Funktion ζ wie folgt:**

  ζ: $\{0,1\}^n$ x $\{0,1\}^n$ → {true, false} mit

  ζ(x,x‘) = true genau dann, wenn ∃ (s, s‘)∈ →: code(x)=s ∧ code(x‘)=s‘

  salopp: „ ζ(x,x‘) ist genau dann wahr, wenn x bzw. x‘ Kodierungen von aufeinander folgenden Zuständen sind.“

- **Schritt 3: Stelle die boolesche Funktion ζ als aussagenlogische Formel dar.**

  Ergibt sich recht trivial bei der Konstruktion von ζ bei Schritt 2.

Symbolische Darstellung der Kripke-Struktur aus Beisp. Folie 6

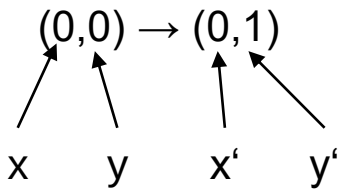
Durchführung

- **Schritt 1: Kodiere die Zustände als Bitvektoren**

  $$\text{code}(s_0)=(0,0),\ \text{code}(s_1)=(0,1),\ \text{code}(s_2)=(1,0)$$

- **Schritt 2/3: Transition als boolesche Funktion**

  Betrachte Übergang $s_0 \rightarrow s_1$. Dies entspricht:



$$\neg x \wedge \neg y \wedge \neg x' \wedge y'$$

Kodiert man alle Übergänge, so erhält man für ζ die aussagenlogische Formel:

$$\zeta = (\neg x \wedge \neg y \wedge \neg x' \wedge y') \vee (\neg x \wedge y \wedge \neg\, y') \vee (x \wedge \neg y \wedge \neg\, y')$$

Zusammenfassung

- Model Checking ist eine automatisierte Überprüfung von Systembeschreibungen gegen eine Spezifikation von Systemeigenschaften.
- Sowohl die Systembeschreibung als auch die Spezifikation wird dem Model-Checker als Input übergeben.
- Als mathematisches Modell für die Systembeschreibung werden beispielsweise Kripke-Strukturen verwendet.
- Spezifikationen der Modelleigenschaften lassen sich mittels temporaler Logiken spezifizieren.
- Man unterscheidet generell beim Model Checking zwischen „explizitem“ und „symbolischem“ Model-Checking.
- Die Ergebnisse der Verifikation gelten nur für das Modell und dürfen nicht bedenkenlos auf das reale System übertragen werden.

Literatur:

- Christel Baier, Joost-Pieter Katoen: Principles of Model Checking, MIT Press, 2008. (Standardwerk)
- Fred Kröger: Temporale Logik und Zustandssysteme, Kurzskriptum LMU, 1994. (www.pst.ifi.lmu.de/lehre/WS0405/tl/skriptum/skriptum.ps; Abruf am 02.12.13)
- 6-Day Course on Principles of Model Checking, University of Twente, Enschede, September 29 – October 2, and October 8+9, 2009
(Kursunterlagen: http://www-i2.informatik.rwth-aachen.de/i2/principles/ ; Abruf am 02.12.13)