

Formale Spezifikation und Verifikation

Wintersemester 2024

Prof. Dr. Gidon Ernst

gidon.ernst@lmu.de

Software and Computational Systems Lab
Ludwig-Maximilians-Universität München, Germany

September 30, 2024



Modellierung/Spezifikation reativer Systeme

Modellierung/Spezifikation reativer Systeme

Beispiele zur Modellierung/Spezifikation

Ein “Webserver”

```
from socket import *
from _thread import *

addr = ('', 80)
s = socket(AF_INET,
           SOCK_STREAM)
s.bind(addr)
s.listen()

while True:
    conn = s.accept()
    start_new_thread(serve, conn)
```

Ein “Webserver”

```
from socket import *
from _thread import *

addr = ('', 80)
s = socket(AF_INET,
           SOCK_STREAM)
s.bind(addr)
s.listen()

while True:
    conn = s.accept()
    start_new_thread(serve, conn)
```

Gewünschte Eigenschaften

- ▶ Kein Absturz aber auch keine (selbstständige) Terminierung
- ▶ Server reagiert auf Verbindungsaufbau
- ▶ Anfragen werden irgendwann beantwortet
- ▶ Sicheres Multi-Threading
- ▶ Fairness bezgl Reihenfolge der Verbindungen
- ▶ ...

Ein “Webserver”

```
from socket import *
from _thread import *

addr = ('', 80)
s = socket(AF_INET,
           SOCK_STREAM)
s.bind(addr)
s.listen()

while True:
    conn = s.accept()
    start_new_thread(serve, conn)
```

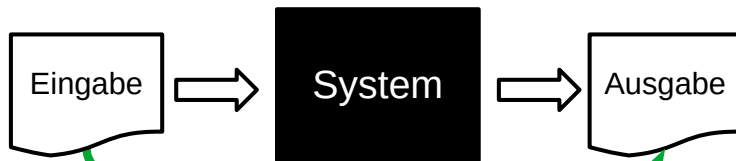
Gewünschte Eigenschaften

- ▶ Kein Absturz aber auch keine (selbstständige) Terminierung
- ▶ Server reagiert auf Verbindungsaufbau
- ▶ Anfragen werden irgendwann beantwortet
- ▶ Sicheres Multi-Threading
- ▶ Fairness bezgl Reihenfolge der Verbindungen
- ▶ ...

Interessantes zeitliches Verhalten!

Nebenläufigkeit, Kommunikation, ...

Traditionelle Softwaresysteme



Spezifikation durch Vor+Nachbedingung

- ▶ Hoare-Logik: beschreibt diese Außensicht
- ▶ Erreichbarkeitsanalyse: Menge der Zwischenzustände
- ▶ Heute *Zeitliche Abfolge* von Zuständen/Ereignissen

Reaktive, Eingebettete Systeme



```
int sqliteProcessJoin(Parse *pParse, Select *p){
    SrcList_item *pLeft; /* Left table being joined */
    SrcList_item *pRight; /* Right table being joined */

    p = p->pSrc;
    pLeft = p->a[0];
    pRight = p->a[1];
    for(j=0; j<p->nSrc; j++){
        if( pLeft->nCol == pRight->nCol ){
            sqlite3Error(pParse, "a NATURAL join may not have "
                "an ON or USING clause", 0);
            return 1;
        }
        for(j=0; j<pRightTab->nCol; j++){
            char *zName; /* Name of column in the right table */
            int iLeft; /* Matching left table */
            int iLeftCol; /* Matching column in the left table */

            zName = pRightTab->aCol[j].zName;
            if( tableAndColumnIndex(pSrc, i+1, zName, &iLeftCol)
```



Agenda

Ziel: Grundverständnis für reaktive Systeme entwickeln

- ▶ Modellierung: Transitionssysteme mit Ereignissen/Ein-/Ausgaben)
- ▶ Spezifikation: Lineare Temporale Logik (LTL)
- ▶ Sicherheits-/Lebendigkeitseigenschaften

Ausblick:

- ▶ Nebenläufigkeit, Parallelität, evtl verteilte Algorithmen
- ▶ Formale Semantik von LTL
- ▶ Büchi-Automaten, Verifikation (falls Zeit)

Reaktive, Eingebettete Systeme

Eingebettet: Teil einer physikalischen Umgebung
(Re)Aktiv: Ereignisse und Kommunikation

Beispiele: Kaffeemaschinen, Ampelsteuerungen, Geldautomaten, Smart-Home, Smart-Car, Software in der Avionik, medizinische Geräte, ...

Charakteristika (in der Praxis)

- ▶ Aktionen und Ereignisse passieren gleichzeitig
- ▶ Komponenten/Systeme fallen aus, Nachrichten gehen verloren, Messdaten sind ungenau, Interaktion nicht vorhersagbar
- ▶ Konflikt: sicherer vs. reibungsloser Betrieb

Reaktive, Eingebettete Systeme

Eingebettet: Teil einer physikalischen Umgebung
(Re)Aktiv: Ereignisse und Kommunikation

Beispiele: Kaffeemaschinen, Ampelsteuerungen, Geldautomaten, Smart-Home, Smart-Car, Software in der Avionik, medizinische Geräte, ...

Charakteristika (in der Praxis)

- ▶ Aktionen und Ereignisse passieren gleichzeitig
- ▶ Komponenten/Systeme fallen aus, Nachrichten gehen verloren, Messdaten sind ungenau, Interaktion nicht vorhersagbar
- ▶ Konflikt: sicherer vs. reibungsloser Betrieb

Herausforderung für formale Ansätze

- ▶ Präzise Beschreibung ungenauen Verhaltens
- ▶ Zustandsexplosion bei nebenläufigem Verhalten

Erinnerung: Modelle und Abstraktion

- ▶ Spezifikation S heute: Temporallogik
- ▶ Modell M_1 heute: Transitionssysteme/Automaten
- ▶ Modell M_2 (z.B. UML)
- ▶ \vdots
- ▶ Implementierung I (Code, Schaltpläne)

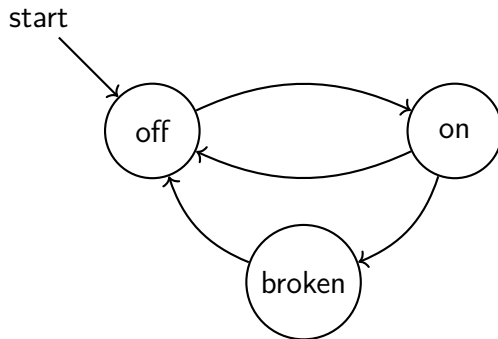
Abstraktion ↑

↓ Verfeinerung

Modellierung/Spezifikation reativer Systeme

Beispiele zur Modellierung

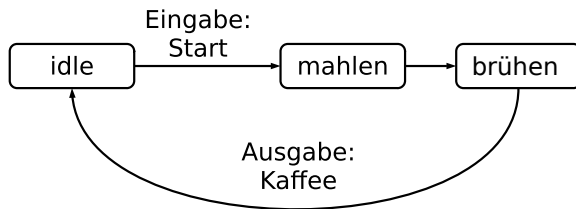
Modellierung: Beamer mit Defekten



Modellierung: Transitionssystem

Abstraktion: es wurde nicht modelliert, wann der Beamer ein/ausgeschaltet wird, und wodurch/wann ein Defekt auftritt, und wie ein solcher repariert wird

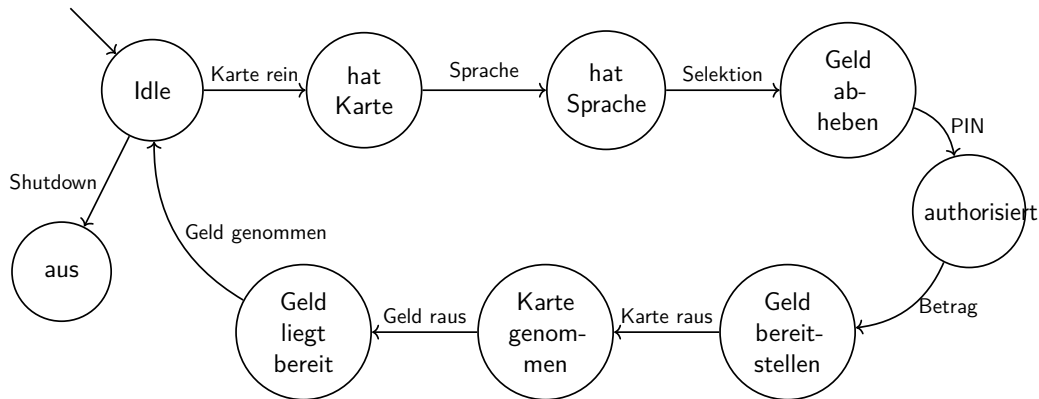
Modellierung: Kaffeemaschine



Modellierung: *markiertes* Transitionssystem
(auch endlicher Automat genannt)

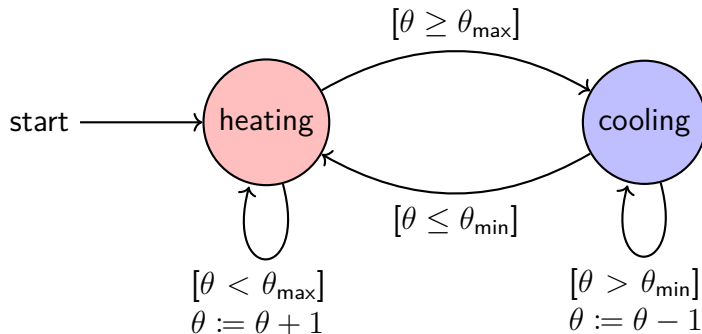
Abstraktion: woher/wann kommt die Eingabe, wieviel/welcher Kaffee?

Modellierung: Geldautomat



Abstraktion: nur (manche) *Interaktionen* modelliert,
z.B. Sprache, Beträge, PIN, ... nicht explizit betrachtet

Modellierung: Thermostat



Modellierung: Automat mit Variablen (\approx Kontrollflussautomat)

Variante: Modellierung als hybrides System:
kontinuierliche Temperaturänderung $\Delta t > 0$ und $\Delta t < 0$

Nebenläufige Programme

$x = 0, y = 0$

$\{\mathbf{while} \ x < 2 \wedge y < 2 \ \mathbf{do} \ x++\} \parallel \{\mathbf{while} \ x < 2 \wedge y < 2 \ \mathbf{do} \ y++\}$

Nebenläufige Programme

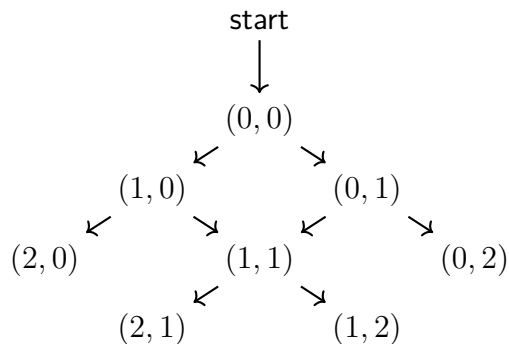
$x = 0, y = 0$

$\{\text{while } x < 2 \wedge y < 2 \text{ do } x++\} \parallel \{\text{while } x < 2 \wedge y < 2 \text{ do } y++\}$

Zugehöriges Transitionssystem mit Zuständen (x, y)

durch explizite Erreichbarkeitsanalyse berechnet

(abstrahiert: Schleifentest + Zuweisung ein atomarer Schritt)



Nebenläufige Programme

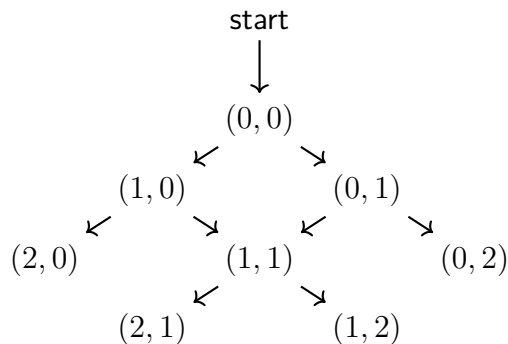
$x = 0, y = 0$

$\{\text{while } x < 2 \wedge y < 2 \text{ do } x++\} \parallel \{\text{while } x < 2 \wedge y < 2 \text{ do } y++\}$

Zugehöriges Transitionssystem mit Zuständen (x, y)

durch explizite Erreichbarkeitsanalyse berechnet

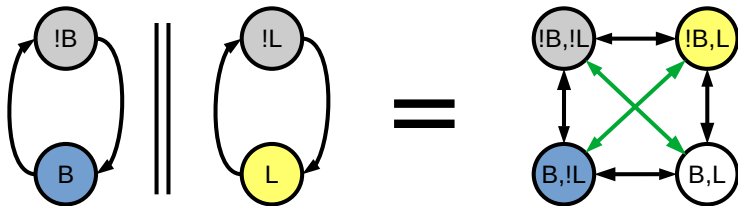
(abstrahiert: Schleifentest + Zuweisung ein atomarer Schritt)



- ▶ Exponentiell viele Kombinationen: welcher Thread (links/rechts) führt den nächsten Schritt aus?
- ▶ Wichtig: Zusammenführen identischer Zustände, hier $(1, 1)$

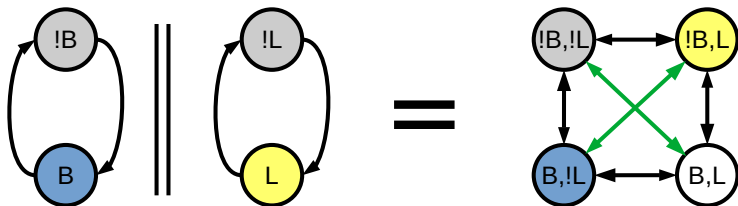
Parallelkomposition von (Transitions-)Systemen

Vorlesungssaal mit Beamer und Licht



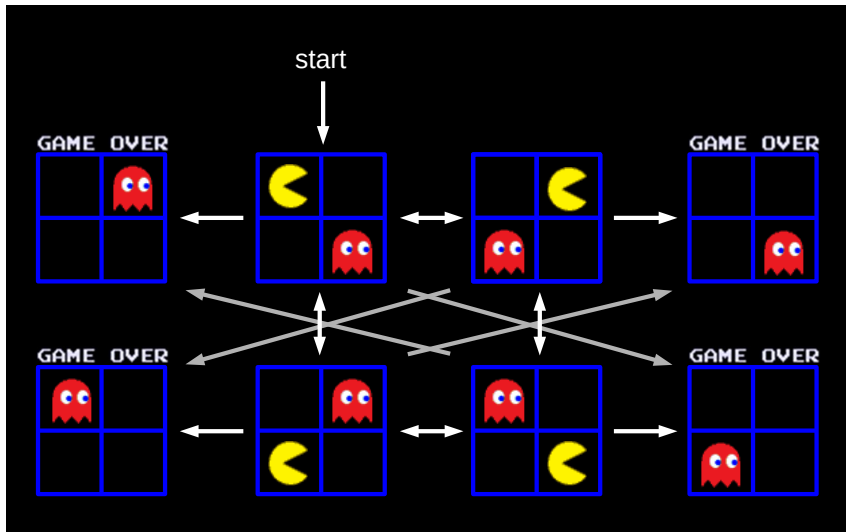
Parallelkomposition von (Transitions-)Systemen

Vorlesungssaal mit Beamer und Licht



- ▶ Zustandsräume durch kartesisches Produkt kombiniert:
 $\Sigma = \Sigma_1 \times \Sigma_2$ mit Zustandsparen (s_1, s_2) sodass $s_1 \in \Sigma_1, s_2 \in \Sigma_2$
- ▶ Designentscheidung: sind **gleichzeitige** Transitionen möglich?
Wie ist die Transitionsrelation \rightarrow mit \rightarrow_1 und \rightarrow_2 definiert?
(Kommunikation = Synchronisation von Transitionen)



Pacman als Transitionssystem



Codierte Regel: Beide (Pacman/Geist) müssen sich in jedem Spielschritt bewegen



Zusammenfassung: Modellierung mit Transitionssystemen

Definition möglicher Zustände

- ▶ Teilzustände identifizieren, z.B.:
Licht an/aus, Beamer an/aus, Positionen von   + game over?)
- ▶ Festlegen der sinnvollen Kombination von Teilzuständen, z.B.:
alle $\{(!B, !L), (B, !L), (!B, L), (B, L)\}$ (Kreuzprodukt),
oder Auswahl: $\{(!B, !L), (!B, L), (B, L)\}$ (Beamer ist an Licht gekoppelt)

Zusammenfassung: Modellierung mit Transitionssystemen

Definition möglicher Zustände



- ▶ Teilzustände identifizieren, z.B.:
Licht an/aus, Beamer an/aus, Positionen von ,  + game over?)
- ▶ Festlegen der sinnvollen Kombination von Teilzuständen, z.B.:
alle $\{(!B, !L), (B, !L), (!B, L), (B, L)\}$ (Kreuzprodukt),
oder Auswahl: $\{(!B, !L), (!B, L), (B, L)\}$ (Beamer ist an Licht gekoppelt)

Definition möglicher Transitionen

- ▶ Aus welchem Zustand kann in welchen gewechselt werden?

Zusammenfassung: Modellierung mit Transitionssystemen

Definition möglicher Zustände

- ▶ Teilzustände identifizieren, z.B.:
Licht an/aus, Beamer an/aus, Positionen von   + game over?)
- ▶ Festlegen der sinnvollen Kombination von Teilzuständen, z.B.:
alle $\{(!B, !L), (B, !L), (!B, L), (B, L)\}$ (Kreuzprodukt),
oder Auswahl: $\{(!B, !L), (!B, L), (B, L)\}$ (Beamer ist an Licht gekoppelt)

Definition möglicher Transitionen

- ▶ Aus welchem Zustand kann in welchen gewechselt werden?

Visualisierung als Graph (wie in den Folien, z.B. für die Übung)

Alternativ: Mengenschreibweise für $T = (\Sigma, \sigma^I, \rightarrow)$, z.B.

- ▶ $\Sigma = \{\text{on}, \text{off}\}$ mit $\sigma^I = \{\text{off}\}$
- ▶ $\rightarrow = \{(\text{on}, \text{off}), (\text{off}, \text{on})\}$ oder $\text{on} \rightarrow \text{off}, \text{off} \rightarrow \text{on}$

Modellierung/Spezifikation reativer Systeme

Beispiele zur Spezifikation

Motivation

Vormals: explizite/symbolischen Erreichbarkeit

- ▶ Analyse: Berechnung der erreichbaren Zustände σ^R
- ▶ Spezifikation: Formel ϕ über *Zustände*
- ▶ Verifikation: Prüfe für alle $s \in \sigma^R$ ob $s \models \phi$ (ϕ gilt in s)

Vormals: Floyd/Hoare Logik (Vor-/Nachbedingungen)

Motivation

Vormals: explizite/symbolischen Erreichbarkeit

- ▶ Analyse: Berechnung der erreichbaren Zustände σ^R
- ▶ Spezifikation: Formel ϕ über *Zustände*
- ▶ Verifikation: Prüfe für alle $s \in \sigma^R$ ob $s \models \phi$ (ϕ gilt in s)

Vormals: Floyd/Hoare Logik (Vor-/Nachbedingungen)

Neu: Eigenschaften über *zeitliche Zusammenhänge*

- ▶ Spezifikation: Formeln ϕ in *Temporalen Logik*
- ▶ Bedeutung: $\bar{s} \models \phi$ wird über ganzen Spuren ausgewertet
- ▶ Allgemein: \bar{s} auch *unendliche* Abläufe

Motivation

Vormals: explizite/symbolischen Erreichbarkeit

- ▶ Analyse: Berechnung der erreichbaren Zustände σ^R
- ▶ Spezifikation: Formel ϕ über *Zustände*
- ▶ Verifikation: Prüfe für alle $s \in \sigma^R$ ob $s \models \phi$ (ϕ gilt in s)

Vormals: Floyd/Hoare Logik (Vor-/Nachbedingungen)

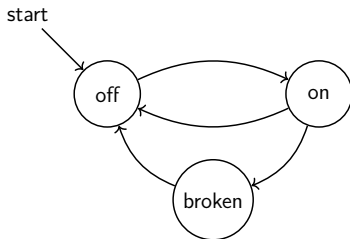
Neu: Eigenschaften über *zeitliche Zusammenhänge*

- ▶ Spezifikation: Formeln ϕ in *Temporalen Logik*
- ▶ Bedeutung: $\bar{s} \models \phi$ wird über ganzen Spuren ausgewertet
- ▶ Allgemein: \bar{s} auch *unendliche* Abläufe
- ▶ Beispiele:
 - $\Box \phi$ ϕ gilt immer (entspricht $\forall s \in \sigma^R. s \models \phi$)
 - $\Diamond (x = 0)$ in der Zukunft gilt irgendwann einmal $x = 0$
 - $P \mathcal{U} Q$ P gilt eine Weile bis garantiert Q eintritt

Syntax der Linearen Temporale Logik (LTL)

Formeln $\phi ::=$	$\text{true} \mid \text{false} \mid A$	atomare Propositionen
	$\mid \neg\phi \mid \phi \wedge \psi \mid \dots$	Formeln mit Junktoren
	$\mid \circ \phi \quad (\mathcal{N} \phi)$	ϕ gilt im <i>nächsten</i> Zustand “next” ϕ
	$\mid \Diamond \phi \quad (\mathcal{F} \phi)$	ϕ gilt <i>irgendwann</i> jetzt oder später “eventually” ϕ (auch “finally”)
	$\mid \Box \phi \quad (\mathcal{G} \phi)$	ϕ gilt ab jetzt <i>immer</i> “always” ϕ (auch “globally”)
	$\mid \phi \mathcal{U} \psi$	ϕ gilt bis irgendwann ψ garantiert eintritt ϕ “until” ψ
	$\mid \phi \mathcal{W} \psi$	ϕ gilt solange bis ψ vielleicht eintritt ϕ “weak until” ψ

Spezifikation: Beamer mit Defekten



Welche Eigenschaften gelten? Wir verfolgen Ausführungen vom Startzustand aus:

✗ on

jetzt (im Startzustand) on

✓ \diamond on

irgendwann on

✗ \square off

immer off

✗ $\text{broken} \wedge \circ \text{off}$

jetzt broken und im *nächsten* Zustand off

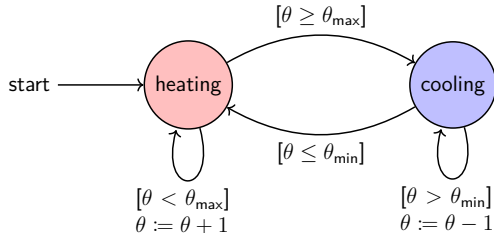
✓ $\square(\text{broken} \rightarrow \circ \text{off})$

immer: falls jetzt broken, dann danach off

✓ $\circ \circ (\text{broken} \vee \text{off})$

im *übernächsten* Zustand off oder broken

Spezifikation: Thermostat



Welche Eigenschaften gelten? Wir verfolgen Ausführungen vom Startzustand aus:

- ✓ $\Box (t_{\min} \leq t \leq t_{\max})$ Temperatur bleibt *immer* in den Grenzen
- ✓ $(\text{heating} \wedge t \leq t_{\max}) \mathcal{U} \text{cooling}$
 - ▶ es wird *zunächst* geheizt und die Temperatur bleibt unter dem Maximum
 - ▶ *danach* kommt *garantiert* die Kühlphase
- ✓ $\Box \Diamond \text{cooling}$ es wird *immer* wieder *irgendwann* gekühlt
- ✗ $\Diamond \Box \text{cooling}$ *irgendwann* wird *nur noch* gekühlt
- ✗ $\circ \text{cooling}$ (nicht notwendigerweise)
- ✓ $\Box (t = t_{\max} \Rightarrow \circ \text{cooling})$

Klassifikation von Eigenschaften

Sicherheitseigenschaften

Ziel

- ▶ “etwas Schlechtes passiert nie”
- ▶ Fehler/Konflikte vermeiden
- ▶ Einhalten von Grenzwerten

Charakteristik: es reicht aus,
alle endlichen Abläufe zu prüfen

Beispiele

- ▶ Hoare-Tripel
- ▶ (Schleifen/Klassen)-Invarianten
- ▶ Wechselseitiger Ausschluss

Klassifikation von Eigenschaften

Sicherheitseigenschaften

Ziel

- ▶ “etwas Schlechtes passiert nie”
- ▶ Fehler/Konflikte vermeiden
- ▶ Einhalten von Grenzwerten

Charakteristik: es reicht aus,
alle endlichen Abläufe zu prüfen

Beispiele

- ▶ Hoare-Tripel
- ▶ (Schleifen/Klassen)-Invarianten
- ▶ Wechselseitiger Ausschluss

Lebendigkeitseigenschaften

Ziel

- ▶ “etwas Gutes passiert irgendwann”
- ▶ Fortschritt erzielen
- ▶ Reaktivität erzwingen

Charakteristik: kann auf endlichen
Abläufen nicht widerlegt werden

Beispiele

- ▶ Terminierung von Programmen
- ▶ Anfragen werden beantwortet
- ▶ Fehlerzustände werden wieder verlassen

Typische Formelmuster

Sicherheitseigenschaften

- ▶ Invariante P gilt immer $\square P$
- ▶ Invariante P zumindest bis Q $P \mathcal{W} Q$ Q muss aber nicht eintreten

Lebendigkeitseigenschaften

- ▶ Garantie P tritt ein $\diamond P$
- ▶ Garantie Q tritt ein $P \mathcal{U} Q$ bis da hin immer P
- ▶ Recurrence/Progress $\square \diamond P$ immer wieder P
- ▶ Response $\square (P \Rightarrow \diamond Q)$ auf jedes P folgt garantiert Q
- ▶ Stability $\diamond \square P$ irgendwann nur noch P

Klassifikation nicht immer offensichtlich

↪ Kann die Eigenschaft schon nach endlich vielen Schritten widerlegt werden?

✓ Sicherheitseigenschaft ✗ Lebendigkeitseigenschaft

LTL: Äquivalenzen

Wir kommen mit \circ und \mathcal{U} aus:

$$\blacktriangleright \Diamond \phi \iff \text{true} \mathcal{U} \phi$$

$$\blacktriangleright \Box \phi \iff \neg \Diamond \neg \phi$$

$$\blacktriangleright \phi \mathcal{W} \psi \iff \phi \mathcal{U} \psi \vee \Box \phi$$

LTL: Äquivalenzen

Wir kommen mit \circ und \mathcal{U} aus:

- ▶ $\Diamond \phi \iff \text{true } \mathcal{U} \phi$
- ▶ $\Box \phi \iff \neg \Diamond \neg \phi$
- ▶ $\phi \mathcal{W} \psi \iff \phi \mathcal{U} \psi \vee \Box \phi$

Außerdem lassen sich alle Operatoren “einen Schritt weiterschieben”

- ▶ $\Box \phi \iff \phi \wedge \circ \Box \phi$ ϕ gilt jetzt und auch später immer
- ▶ $\Diamond \phi \iff \phi \vee \circ \Diamond \phi$ ϕ gilt jetzt oder später irgendwann
- ▶ $\phi \mathcal{U} \psi \iff \psi \vee \phi \wedge \circ (\phi \mathcal{U} \psi)$ ψ jetzt oder ϕ und wieder das selbe später

Was Sie wissen und können sollten

Von heute

- ▶ Einfache Systeme mit Transitionssystemen modellieren können
 - ▶ Beschreibung des Zustandsraumes evtl aus mehreren Teilen
 - ▶ Angabe möglicher Transitionen
- ▶ Spezifikationen in LTL formulieren
- ▶ Grundsätzliche Unterscheidung zwischen Sicherheitseigenschaften und Lebendigkeitseigenschaften kennen

Ausblick

- ▶ Formale Semantik von LTL + Intuition dazu
Insbesondere: Beispiele zu Sicherheit/Lebendigkeit
- ▶ Modellierung: Automaten mit Ein-/Ausgaben
- ▶ Verifikationstechniken für Automaten bezgl. LTL Spezifikationen

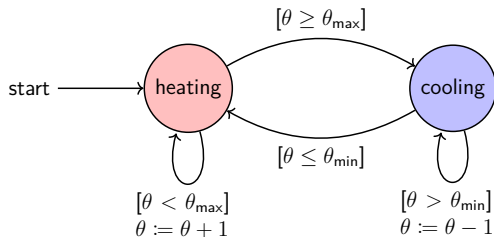
Modellierung/Spezifikation reativer Systeme

Auswertung von LTL Formeln

Auswertung von LTL Formeln auf *Abläufen* von Modellen

- ▶ Intuition, grafische Repräsentation
- ▶ Formale Definition

Thermostat



Modell mit Variable t und (\approx Kontrollflussautomat),
Nehmen wir mal an: $t_{\min} = 19$, $t_{\max} = 23$ konstant

Erinnerung: das zugehörige Transitionssystem beschreibt

- ▶ Zustände, z.B. $s = \{\text{pc} \mapsto \text{heating}, t \mapsto 23\}$ mit konkreten Werten
 $s' = \{\text{pc} \mapsto \text{cooling}, t \mapsto 23\}$
- ▶ Initialzustände, z.B. $s_0 = \{\text{pc} \mapsto \text{heating}, t \mapsto 20\}$
- ▶ Mögliche Transitionen, z.B. $s \rightarrow s'$

Definition: Eine *unendliche* Folge $\bar{s} = (s_0, s_1, \dots)$ von Zuständen $s_i \in \Sigma$ ist *Ablauf* eines Transitionssystems $T = (\Sigma, \sigma^I, \rightarrow)$ falls

- ▶ $s_0 \in \sigma^I$ ist Initialzustand
- ▶ $s_i \rightarrow s_{i+1}$ gemäß möglicher Transitionen

Definition: Eine *unendliche* Folge $\bar{s} = (s_0, s_1, \dots)$ von Zuständen $s_i \in \Sigma$ ist *Ablauf* eines Transitionssystems $T = (\Sigma, \sigma^I, \rightarrow)$ falls

- ▶ $s_0 \in \sigma^I$ ist Initialzustand
- ▶ $s_i \rightarrow s_{i+1}$ gemäß möglicher Transitionen

Grafische Darstellung



Definition: Eine *unendliche* Folge $\bar{s} = (s_0, s_1, \dots)$ von Zuständen $s_i \in \Sigma$ ist *Ablauf* eines Transitionssystems $T = (\Sigma, \sigma^I, \rightarrow)$ falls

- ▶ $s_0 \in \sigma^I$ ist Initialzustand
- ▶ $s_i \rightarrow s_{i+1}$ gemäß möglicher Transitionen

Grafische Darstellung



Spezifikation von Eigenschaften

- ▶ über einzelne Zustände, z.B. gilt $t \geq t_{\min}$ in s_0 ?
- ▶ über alle *erreichbaren* Zustände, z.B. $\forall s \in \sigma^R. s \models t \geq t_{\min}$?

Definition: Eine *unendliche* Folge $\bar{s} = (s_0, s_1, \dots)$ von Zuständen $s_i \in \Sigma$ ist *Ablauf* eines Transitionssystems $T = (\Sigma, \sigma^I, \rightarrow)$ falls

- ▶ $s_0 \in \sigma^I$ ist Initialzustand
- ▶ $s_i \rightarrow s_{i+1}$ gemäß möglicher Transitionen

Grafische Darstellung



Spezifikation von Eigenschaften

- ▶ über einzelne Zustände, z.B. gilt $t \geq t_{\min}$ in s_0 ?
- ▶ über alle *erreichbaren* Zustände, z.B. $\forall s \in \sigma^R. s \models t \geq t_{\min}$?
- ▶ über *gesamte Abläufe* \rightsquigarrow **Lineare Temporeale Logik**

Wiederholung: LTL Syntax

Formeln $\phi ::= P$

| $\neg\phi$ | $\phi \wedge \psi$ | ...

| $\circ \phi$ ($\mathcal{N} \phi$)

| $\Diamond \phi$ ($\mathcal{F} \phi$)

| $\Box \phi$ ($\mathcal{G} \phi$)

| $\phi \mathcal{U} \psi$

| $\phi \mathcal{W} \psi$

Zustandsformeln

Junktoren über LTL Formeln

ϕ gilt im *nächsten* Zustand

ϕ gilt *irgendwann* jetzt oder später

ϕ gilt ab jetzt *immer*

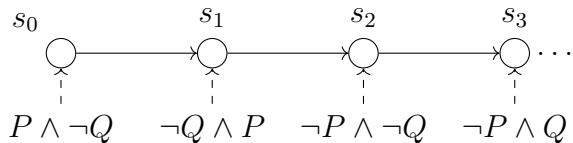
ϕ gilt bis irgendwann ψ garantiert eintritt

ϕ gilt solange bis ψ vielleicht eintritt

Auswertung von Formeln (grafisch)

Gegeben: Auswertung $s \models P$ von Zustandsformeln (“normale Formeln”)

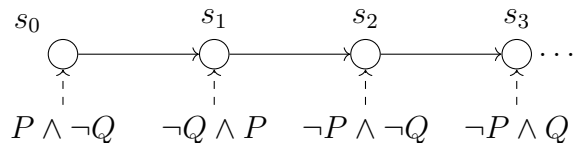
Zustandsformeln können zu *unterschiedlichen Zeitpunkten* in einem Ablauf gelten.
Grafische Darstellung (Beispiel):



Auswertung von Formeln (grafisch)

Gegeben: Auswertung $s \models P$ von Zustandsformeln (“normale Formeln”)

Zustandsformeln können zu *unterschiedlichen Zeitpunkten* in einem Ablauf gelten.
Grafische Darstellung (Beispiel):



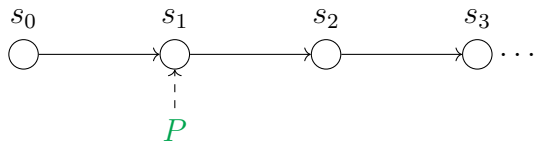
Ziel heute

- ▶ Grafische Auswertung (Bedeutung/Semantik) einfacher LTL Formeln
- ▶ Formale Definition der Auswertung $\bar{s} \models \phi$ für beliebig geschachtelte LTL

Semantik von “next” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\circ P$ wahr wird?

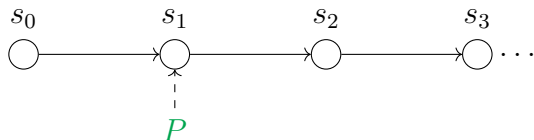
✓ $\circ P$



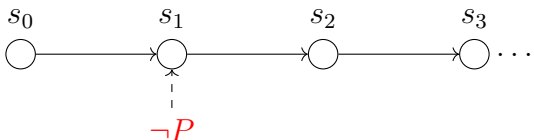
Semantik von “next” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\circ P$ wahr wird?

✓ $\circ P$



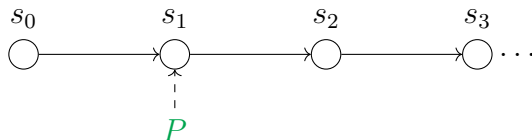
✗ $\circ P$



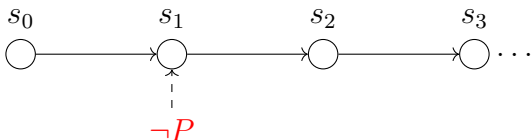
Semantik von "next" grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\circ P$ wahr wird?

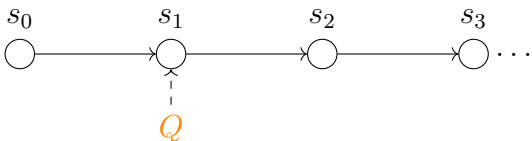
✓ $\circ P$



✗ $\circ P$

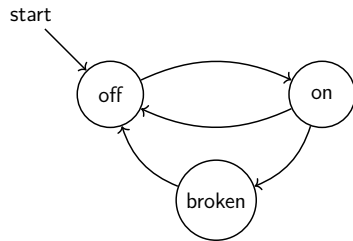


? $\circ P$



Im letzten Beispiel ist keine Information über $s_1 \models P$ angegeben

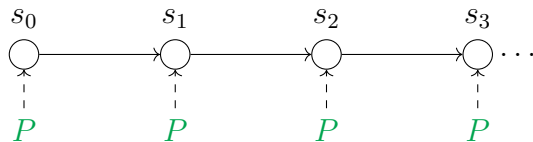
Spezifikation: Beamer mit Defekten



Semantik von “always” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Box P$ wahr wird?

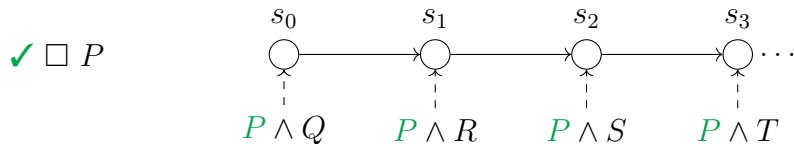
✓ $\Box P$



sofern für alle weiteren Zustände $s_k \models P$ gilt

Semantik von “always” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Box P$ wahr wird?

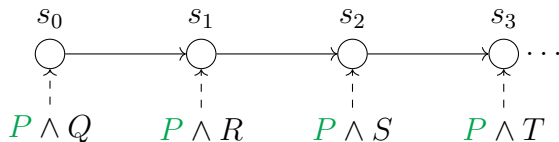


sofern für alle weiteren Zustände $s_k \models P$ gilt

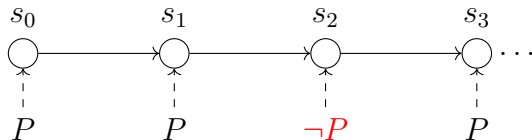
Semantik von “always” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Box P$ wahr wird?

✓ $\Box P$



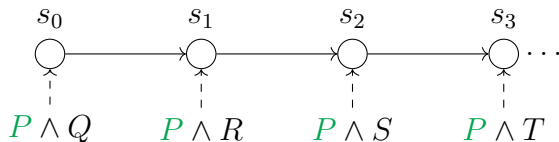
✗ $\Box P$



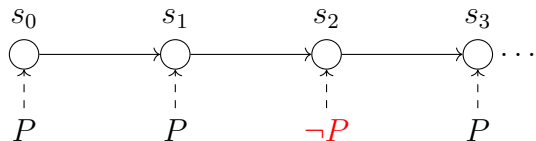
Semantik von “always” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Box P$ wahr wird?

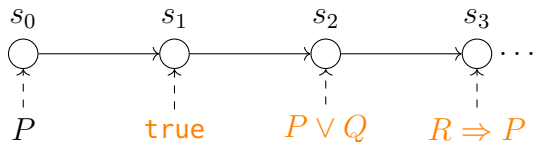
✓ $\Box P$



✗ $\Box P$

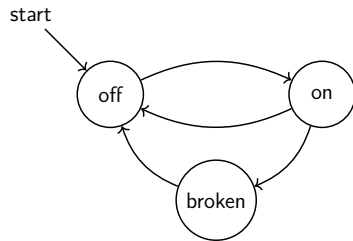


? $\Box P$



Im letzten Beispiel: für s_1, s_2, s_3 ist nicht angegeben ob dort P garantiert gilt

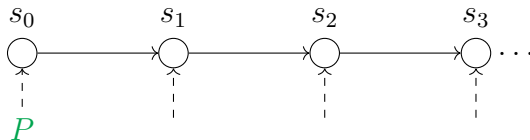
Spezifikation: Beamer mit Defekten



Semantik von “eventually” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Diamond P$ wahr wird?

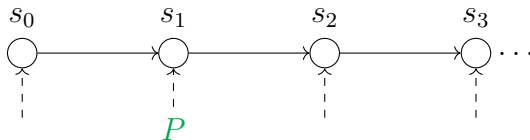
✓ $\Diamond P$



Semantik von “eventually” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Diamond P$ wahr wird?

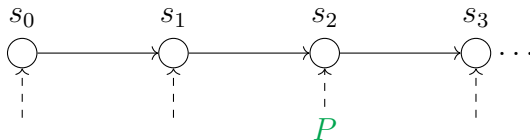
✓ $\Diamond P$



Semantik von “eventually” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Diamond P$ wahr wird?

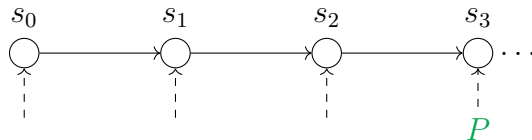
✓ $\Diamond P$



Semantik von “eventually” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Diamond P$ wahr wird?

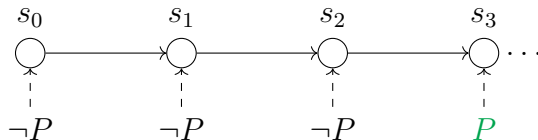
✓ $\Diamond P$



Semantik von “eventually” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Diamond P$ wahr wird?

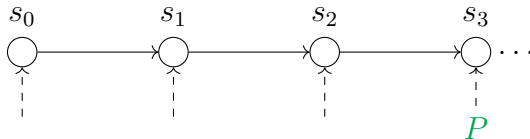
✓ $\Diamond P$



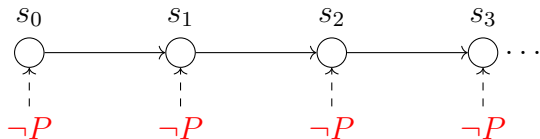
Semantik von “eventually” grafisch

Schematisch: wo muss P gelten damit die LTL Formel $\Diamond P$ wahr wird?

✓ $\Diamond P$

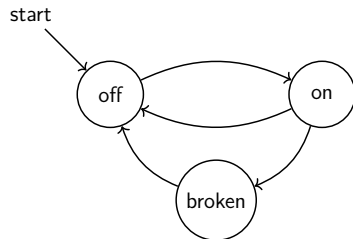


✗ $\Diamond P$



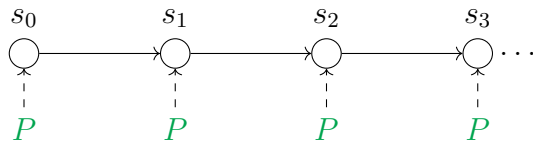
sofern nie $s_k \models P$ gilt für ein $k \geq 0$

Spezifikation: Beamer mit Defekten



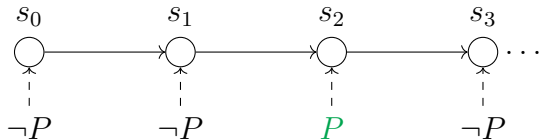
Unendliche Abläufe: Gültigkeit von “always” und “eventually”

✓ $\square P$



- ▶ Um eine Formel $\square P$ zu *beweisen*, muss P auf dem *ganzen* Ablauf gelten
- ▶ Es müssen *alle* Zustände geprüft werden, also unendlich viele

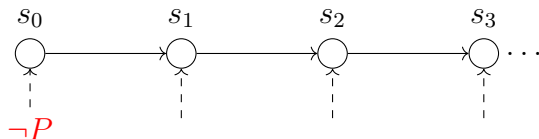
✓ $\diamond P$



- ▶ Um eine Formel $\diamond P$ zu *belegen*, reicht *ein* passender Zustand
- ▶ Sobald P wahr wird, kann man abbrechen

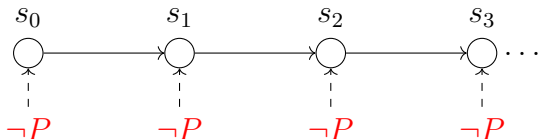
Unendliche Abläufe: Widerlegung von “always” und “eventually”

✗ $\Box P$



- ▶ Um eine Formel $\Box P$ zu *widerlegen*, reicht ein Zustand mit $\neg P$ (hier: s_0)
- ▶ Sobald dieser gefunden ist, kann man abbrechen

✗ $\Diamond P$



- ▶ Um eine Formel $\Diamond P$ zu *widerlegen*, muss man beweisen, dass P nie wahr wird
- ▶ Es müssen analog zu $\checkmark \Box$ unendlich viele Zustände berücksichtigt werden

Diskussion: unendliche Abläufe

Erinnerung: Reaktive Systeme

- ▶ Nichtterminierung gewünscht (z.B. Server, Ampel)
- ▶ Ereignisse treten *immer wieder* auf

Diskussion: unendliche Abläufe

Erinnerung: Reaktive Systeme

- ▶ Nichtterminierung gewünscht (z.B. Server, Ampel)
- ▶ Ereignisse treten *immer wieder* auf

Sicherheitseigenschaft, z.B. $\Box P$

- ▶ “etwas Schlechtes passiert nie”
- ▶ Sofern die Eigenschaft nicht nach einer (beliebigen) endlichen Anzahl von Schritten verletzt werden kann, gilt sie definitiv auf jedem unendlichen Ablauf

Sicherheitseigenschaft, z.B. $\Diamond P$

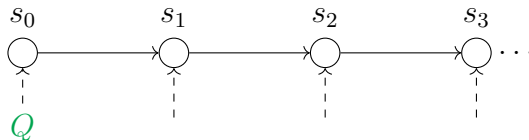
- ▶ “etwas Gutes passiert irgendwann”
- ▶ Sofern man nicht den ganzen Ablauf betrachtet, kann es ja sein, dass P doch noch wahr wird
- ▶ Nicht in endlicher Zeit definitiv widerlegbar

Semantik von “until” und “weak until” grafisch

Schematisch: wo muss/kann P, Q gelten damit $P \mathcal{U} Q$ wahr wird?

✓ $P \mathcal{U} Q$

✓ $P \mathcal{W} Q$

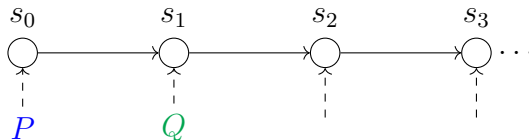


Semantik von “until” und “weak until” grafisch

Schematisch: wo muss/kann P, Q gelten damit $P \mathcal{U} Q$ wahr wird?

✓ $P \mathcal{U} Q$

✓ $P \mathcal{W} Q$

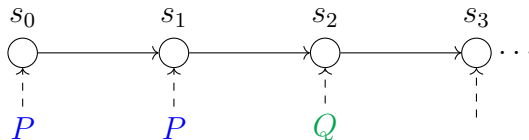


Semantik von “until” und “weak until” grafisch

Schematisch: wo muss/kann P, Q gelten damit $P \mathcal{U} Q$ wahr wird?

✓ $P \mathcal{U} Q$

✓ $P \mathcal{W} Q$

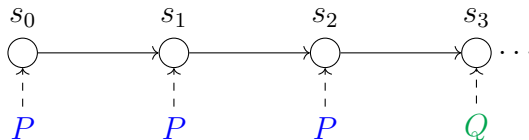


Semantik von “until” und “weak until” grafisch

Schematisch: wo muss/kann P, Q gelten damit $P \mathcal{U} Q$ wahr wird?

✓ $P \mathcal{U} Q$

✓ $P \mathcal{W} Q$

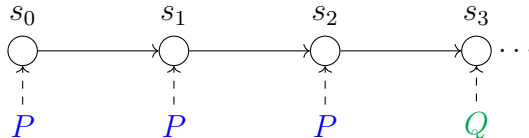


Semantik von “until” und “weak until” grafisch

Schematisch: wo muss/kann P, Q gelten damit $P \mathcal{U} Q$ wahr wird?

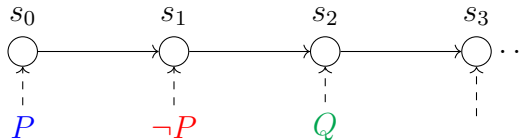
✓ $P \mathcal{U} Q$

✓ $P \mathcal{W} Q$



✗ $P \mathcal{U} Q$

✗ $P \mathcal{W} Q$

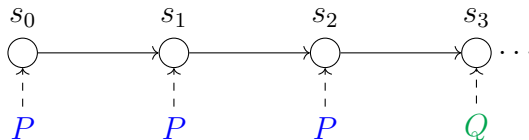


Semantik von “until” und “weak until” grafisch

Schematisch: wo muss/kann P, Q gelten damit $P \mathcal{U} Q$ wahr wird?

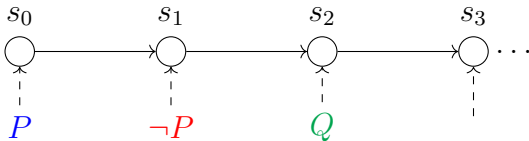
✓ $P \mathcal{U} Q$

✓ $P \mathcal{W} Q$



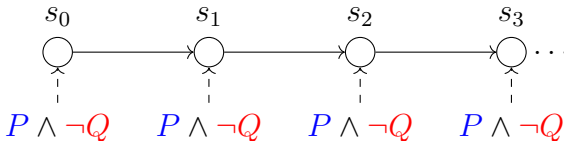
✗ $P \mathcal{U} Q$

✗ $P \mathcal{W} Q$



✗ $P \mathcal{U} Q$

✓ $P \mathcal{W} Q$

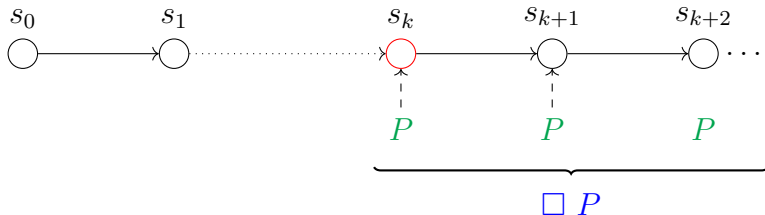


Zwischenstand

- ✓ Auswertung von einfachen LTL Formeln (ohne geschachtelte Operatoren)
- ✓ Sicherheit und Lebendigkeit auf unendlichen Abläufen
- ▶ Wie funktioniert das Schachteln von temporalen Operatoren?

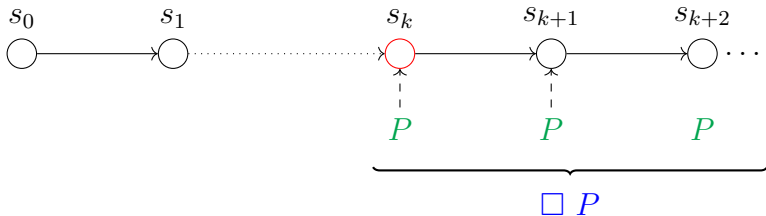
Wie geht man mit geschachtelten Formeln um?

✓ ◇ □ P

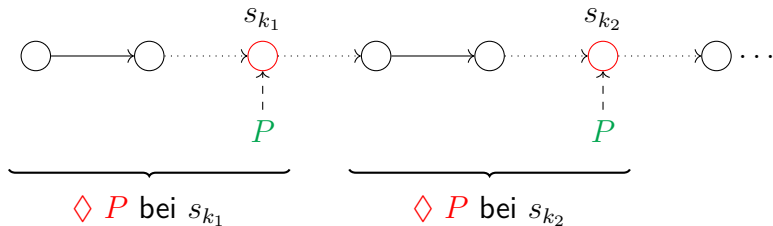


Wie geht man mit geschachtelten Formeln um?

✓ ◇ □ P



✓ □ ◇ P



Formale Semantik von LTL

Definition *Suffix* eines Ablaufes $\bar{s} = (s_0, s_1, \dots)$ ab k

► $\bar{s}|_k = (s_k, s_{k+1}, s_{k+2}, \dots)$ (insbes. $\bar{s}|_0 = \bar{s}$)

$\bar{s} \models P$ gdw $s_0 \models P$

$\bar{s} \models \neg\phi$ gdw $\bar{s} \not\models \phi$ (andere Junktoren analog)

$\bar{s} \models \phi \Rightarrow \psi$ gdw $\bar{s} \not\models \phi$ oder $\bar{s} \models \psi$

Formale Semantik von LTL

Definition *Suffix* eines Ablaufes $\bar{s} = (s_0, s_1, \dots)$ ab k

► $\bar{s}|_k = (s_k, s_{k+1}, s_{k+2}, \dots)$ (insbes. $\bar{s}|_0 = \bar{s}$)

$$\bar{s} \models P$$

$$\text{gdw } s_0 \models P$$

$$\bar{s} \models \neg \phi$$

$$\text{gdw } \bar{s} \not\models \phi \quad (\text{andere Junktoren analog})$$

$$\bar{s} \models \phi \Rightarrow \psi$$

$$\text{gdw } \bar{s} \not\models \phi \text{ oder } \bar{s} \models \psi$$

$$\bar{s} \models \circ \phi$$

$$\text{gdw } \bar{s}|_1 \models \phi$$

$$\bar{s} \models \Box \phi$$

$$\text{gdw } \bar{s}|_k \models \phi \text{ für alle } k \geq 0$$

$$\bar{s} \models \Diamond \phi$$

$$\text{gdw } \bar{s}|_k \models \phi \text{ für mindestens ein } k \geq 0$$

Formale Semantik von LTL

Definition *Suffix* eines Ablaufes $\bar{s} = (s_0, s_1, \dots)$ ab k

► $\bar{s}|_k = (s_k, s_{k+1}, s_{k+2}, \dots)$ (insbes. $\bar{s}|_0 = \bar{s}$)

$\bar{s} \models P$ gdw $s_0 \models P$

$\bar{s} \models \neg\phi$ gdw $\bar{s} \not\models \phi$ (andere Junktoren analog)

$\bar{s} \models \phi \Rightarrow \psi$ gdw $\bar{s} \not\models \phi$ oder $\bar{s} \models \psi$

$\bar{s} \models \circ \phi$ gdw $\bar{s}|_1 \models \phi$

$\bar{s} \models \Box \phi$ gdw $\bar{s}|_k \models \phi$ für alle $k \geq 0$

$\bar{s} \models \Diamond \phi$ gdw $\bar{s}|_k \models \phi$ für mindestens ein $k \geq 0$

$\bar{s} \models \phi \mathcal{U} \psi$ gdw es gibt $k \geq 0$ mit $\bar{s}|_k \models \psi$

und für alle $0 \leq j < k$ gilt $\bar{s}|_j \models \phi$

Formale Semantik von LTL

Definition *Suffix* eines Ablaufes $\bar{s} = (s_0, s_1, \dots)$ ab k

► $\bar{s}|_k = (s_k, s_{k+1}, s_{k+2}, \dots)$ (insbes. $\bar{s}|_0 = \bar{s}$)

$\bar{s} \models P$ gdw $s_0 \models P$

$\bar{s} \models \neg\phi$ gdw $\bar{s} \not\models \phi$ (andere Junktoren analog)

$\bar{s} \models \phi \Rightarrow \psi$ gdw $\bar{s} \not\models \phi$ oder $\bar{s} \models \psi$

$\bar{s} \models \circ \phi$ gdw $\bar{s}|_1 \models \phi$

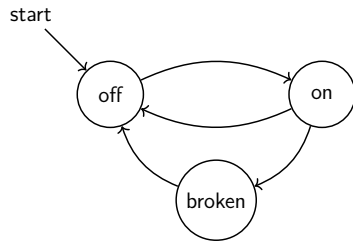
$\bar{s} \models \Box \phi$ gdw $\bar{s}|_k \models \phi$ für alle $k \geq 0$

$\bar{s} \models \Diamond \phi$ gdw $\bar{s}|_k \models \phi$ für mindestens ein $k \geq 0$

$\bar{s} \models \phi \mathcal{U} \psi$ gdw es gibt $k \geq 0$ mit $\bar{s}|_k \models \psi$
und für alle $0 \leq j < k$ gilt $\bar{s}|_j \models \phi$

$\bar{s} \models \phi \mathcal{W} \psi$ gdw $\bar{s} \models \phi \mathcal{U} \psi$ oder $\bar{s} \models \Box \phi$

Spezifikation: Beamer mit Defekten



Spezifikation mit LTL

- ▶ Eine LTL Formel “gilt” (oder nicht) auf einem unendlichen *Ablauf*, der durch ein reaktives System erzeugt wird
- ▶ Temporale Operatoren “verschieben” Teilformeln, sodass Sie sich auf bestimmte Teilabläufe beziehen
- ▶ Ungeschachtelte LTL über Zustandsformeln lässt sich schön grafisch darstellen
- ▶ Geschachtelte Formeln erfordern etwas Nachdenken

© These slides are licensed under the creative commons license:

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0)

- ① give appropriate credit
- ⊖ distribute without modifications
- Ⓜ do not use for commercial purposes