Automatisches Beweisen—Vertiefung
Temporallogik

Christoph Zengler

Arbeitsbereich Symbolisches Rechnen
Prof. Dr. Wolfgang Küchlin
Universität Tübingen

8. Januar 2013

Motivation

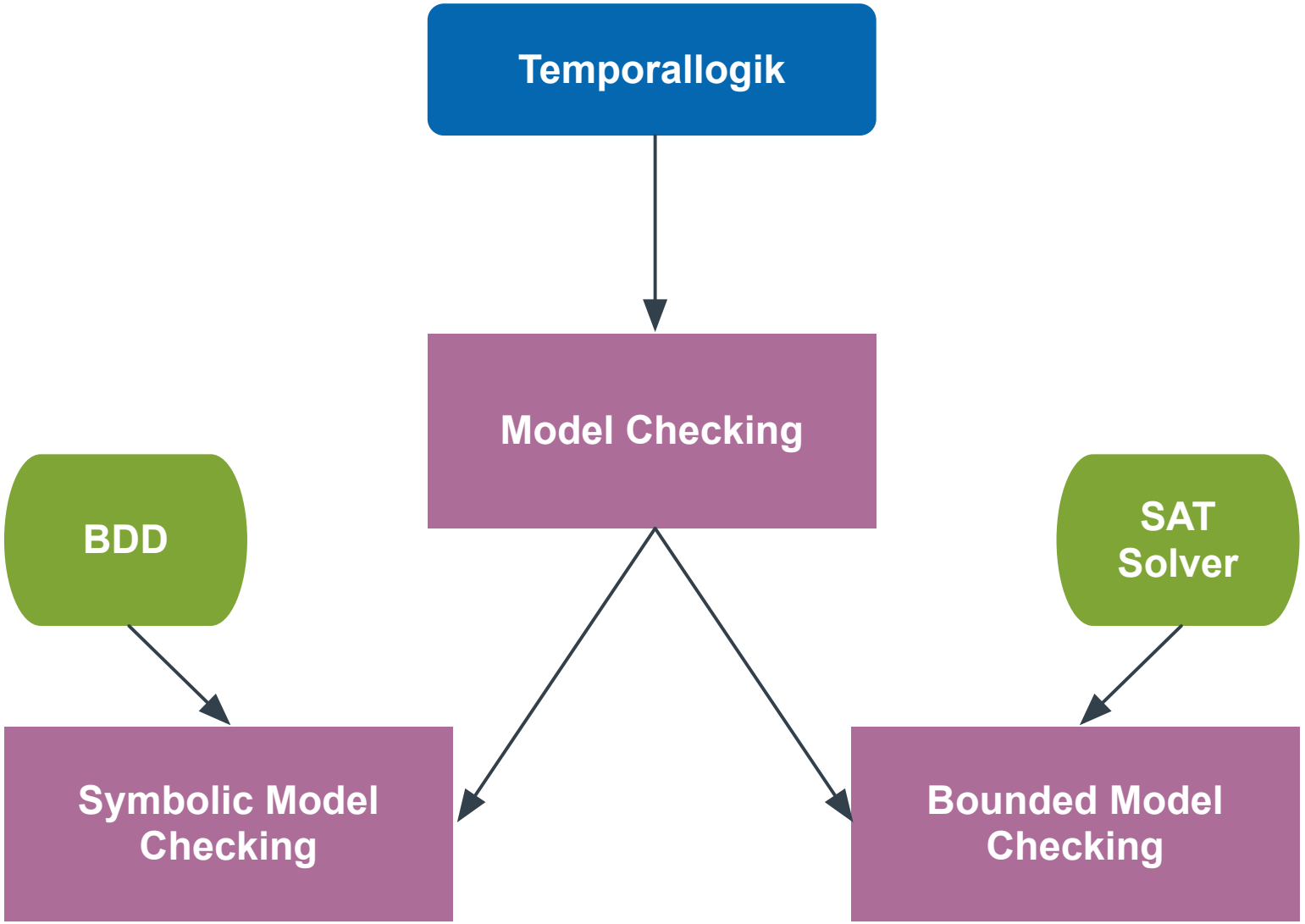
Bisher

- Formeln in AL oder FOL beschreiben einen Zustand
- Was wir nicht modellieren können: Zeitlicher Ablauf
- Modellierung von komplexen Systemen (z.B. Hardware, Software) erfordert jedoch solche Modellierungsmethoden
  - Verschränkung von Threads
  - Ampelschaltung
  - …

$\Downarrow$

Temporallogik

Der Plan für die nächsten drei Wochen



Kripke-Strukturen

- Der Wahrheitswert einer Formel ergibt sich nicht ausschließlich aus der Formel.
- Wir brauchen einen Kontext dazu (zeitliche Abfolge).

$$\Downarrow$$

Menge von Zuständen und Übergänge zwischen diesen Zuständen beschreiben zeitliche Abfolge.

Definition (Kripke-Struktur)

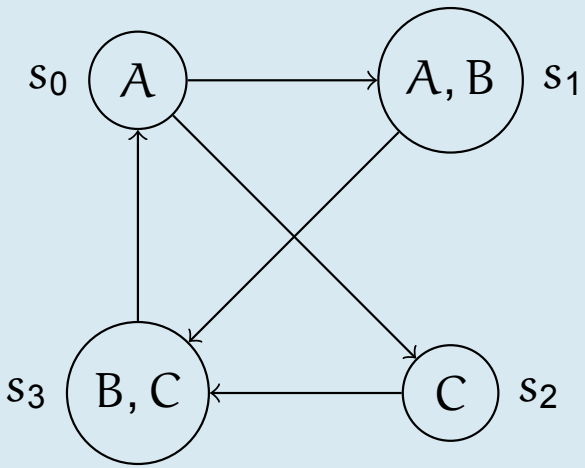
Eine Kripke Struktur ist ein Tripel $\mathcal{M} = (S, \longrightarrow, L)$ mit

- S: Nicht-leere Menge von Zuständen
- $\longrightarrow \subseteq S \times S$: Zustandsübergangsrelation
- L: Beschriftung jedes Zustandes mit dort gültigen aussagenlogischen Variablen

Kripke-Strukturen—Beispiel

Beispiel (Kripke Struktur)

- $S = \{s_0, s_1, s_2, s_3\}$
- $\longrightarrow = \{(s_0, s_1), (s_0, s_2), (s_1, s_3), (s_2, s_3), (s_3, s_0)\}$
- $L : L(s_0) = \{A\}, L(s_1) = \{A, B\}, L(s_2) = \{C\}, L(s_3) = \{B, C\}$



- Anstelle von $(s, t) \in \longrightarrow$ schreiben wir auch $s \longrightarrow t$

Linear Time Logic

- Wir betrachten einzelne Zeitpfade (Pfade innerhalb der Kripke Struktur) auf denen Aussagen irgendwann wahr oder falsch werden
- Es wird immer nur ein Pfad gleichzeitig betrachtet

LTL Operatoren

- $\mathbf{F}\,\varphi$ (Finally): φ gilt in einem zukünftigen Zustand auf dem Pfad
- $\mathbf{G}\,\varphi$ (Globally): φ gilt in allen Zuständen auf dem Pfad
- $\mathbf{X}\,\varphi$ (Next): φ gilt im nächsten Zustand auf dem Pfad
- $\varphi\,\mathbf{U}\,\psi$ (Until): φ gilt, bis ein Zustand erreicht wird, an dem ψ gilt
- $\varphi\,\mathbf{W}\,\psi$ (Weak Until): Wie Until, jedoch muss ψ nie eintreten
- $\varphi\,\mathbf{R}\,\psi$ (Release): ψ gilt bis zu einschließlich dem Zustand, an dem φ erfüllt ist

Syntax

EBNF-Grammatik für LTL Formeln

Formel	=	$\top \mid \bot$	Konstanten
	|	*Variable*	$\in \mathcal{V}$
	|	$\neg$ Formel	Negation
	|	Formel $\wedge$ Formel	Konjunktion
	|	Formel $\vee$ Formel	Disjunktion
	|	Formel $\rightarrow$ Formel	Implikation
	|	$\mathbf{X}$ Formel	Next
	|	$\mathbf{F}$ Formel	Finally
	|	$\mathbf{G}$ Formel	Globally
	|	Formel $\mathbf{U}$ Formel	Until
	|	Formel $\mathbf{W}$ Formel	Weak Until
	|	Formel $\mathbf{R}$ Formel	Release
	|	(Formel)	Klammerung

- Unäre Operatoren binden am stärksten, gefolgt von $\mathbf{R}, \mathbf{U}, \mathbf{W}$, gefolgt von $\wedge, \vee, \rightarrow$.

Semantik

Technische Einschränkung für $\longrightarrow$

- Zu jedem Zustand $s \in S$ gibt es mindestens einen Übergang $s \longrightarrow s'$
- D.h. es gibt keine Deadlocks

i Keine echte Einschränkung

Dies schränkt jedoch die zu modellierenden Systeme nicht ein. Erfüllt ein Zustand s diese Eigenschaft nicht:

- füge einen neuen Zustand s_d (Deadlock) hinzu
- füge $s \longrightarrow s_d$ und $s_d \longrightarrow s_d$ hinzu

Definition (Pfad)

Ein Pfad in einem Modell $\mathcal{M} = (S, \longrightarrow, L)$ ist eine unendliche Folge von Zuständen $s_0, s_1, s_2, \ldots$ in S, so dass für jedes $i \geq 0$ $s_i \longrightarrow s_{i+1}$.

Für einen Pfad $\pi = s_0 \longrightarrow s_1 \longrightarrow \ldots$ schreiben wir π^i für den Teilpfad, der bei s_i beginnt.

Auswertung einer LTL Formel

Algorithmus: $\texttt{holds}(\mathcal{M}, \pi, \psi)$

Eingabe: Modell $\mathcal{M} = (S, \longrightarrow, L)$, Pfad $\pi = s_0 \longrightarrow \ldots$, LTL Formel ψ
Ausgabe: Evaluation von ψ auf dem Pfad π

$\texttt{holds}(\mathcal{M}, \pi, \psi) = \psi \texttt{ match}$

$\quad \top \rightsquigarrow \texttt{true}$

$\quad \mid \bot \rightsquigarrow \texttt{false}$

$\quad \mid \nu \in \mathcal{V} \rightsquigarrow \texttt{if } \nu \in L(s_0) \texttt{ then true else false}$

$\quad \mid \neg\varphi \rightsquigarrow \texttt{if holds}(\mathcal{M}, \pi, \varphi) \texttt{ then false else true}$

$\quad \mid \varphi_1 \wedge \varphi_2 \rightsquigarrow \texttt{if holds}(\mathcal{M}, \pi, \varphi_1) \texttt{ and holds}(\mathcal{M}, \pi, \varphi_2) \texttt{ then true else false}$

$\quad \mid \varphi_1 \vee \varphi_2 \rightsquigarrow \texttt{holds}(\mathcal{M}, \pi, \neg(\neg\varphi_1 \wedge \neg\varphi_2))$

$\quad \mid \varphi_1 \rightarrow \varphi_2 \rightsquigarrow \texttt{holds}(\mathcal{M}, \pi, \neg\varphi_1 \vee \varphi_2))$

$\quad \mid \ldots$

Auswertung einer LTL Formel

Algorithmus: $\texttt{holds}(\mathcal{M}, \pi, \psi)$

$$
\begin{aligned}
&| \ \ldots \\
&| \ \mathbf{X}\,\varphi \ \rightsquigarrow \texttt{holds}(\mathcal{M}, \pi^1, \varphi) \\
&| \ \mathbf{G}\,\varphi \ \rightsquigarrow \ \texttt{for all } i \geq 0\texttt{: holds}(\mathcal{M}, \pi^i, \varphi) \\
&| \ \mathbf{F}\,\varphi \ \rightsquigarrow \ \texttt{exists } i \geq 0\texttt{: holds}(\mathcal{M}, \pi^i, \varphi) \\
&| \ \varphi_1\,\mathbf{U}\,\varphi_2 \ \rightsquigarrow \ \texttt{exists } i \geq 0\texttt{: holds}(\mathcal{M}, \pi^i, \varphi_2) \\
&\qquad\qquad \texttt{and for all } 0 \leq j < i\texttt{: holds}(\mathcal{M}, \pi^j, \varphi_1) \\
&| \ \varphi_1\,\mathbf{W}\,\varphi_2 \ \rightsquigarrow \ \texttt{holds}(\mathcal{M}, \varphi_1\,\mathbf{U}\,\varphi_2)\ \texttt{or} \\
&\qquad\qquad \texttt{for all } k \geq 0\texttt{: holds}(\mathcal{M}, \pi^k, \varphi_1) \\
&| \ \varphi_1\,\mathbf{R}\,\varphi_2 \ \rightsquigarrow \ [\texttt{exists } i \geq 0\texttt{: holds}(\mathcal{M}, \pi^i, \varphi_1)\ \texttt{and for all} \\
&\qquad\qquad 0 \leq j \leq i\texttt{: holds}(\mathcal{M}, \pi^j, \varphi_2)]\ \texttt{or} \\
&\qquad\qquad [\texttt{for all } k \geq 0\texttt{: holds}(\mathcal{M}, \pi^k, \varphi_2)]
\end{aligned}
$$

- Gilt $\texttt{holds}(\mathcal{M}, \pi, \varphi)$, so schreiben wir auch $\pi \vDash_{\mathcal{M}} \varphi$ oder $\pi \vDash \varphi$ wenn $\mathcal{M}$ aus dem Kontext klar ist.
- Gilt $\pi \vDash \varphi$ für jeden möglichen Ausführungspfad π beginnend bei einem Zustand s, so schreiben wir auch $s \vDash \varphi$.

Temporaloperatoren

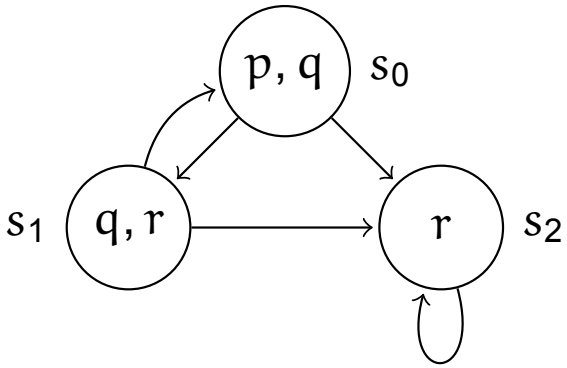
Beispiele

Beispiel (Temporaloperatoren)

	0	1	2	3	4
	○	○	○	○	○
$\mathbf{G}\,p$	p	p	p	p	p
$\mathbf{F}\,p$	-	-	-	p	-
$p\,\mathbf{U}\,q$	p	p	p	q	-
$p\,\mathbf{W}\,q$	p	p	p	q	-
	p	p	p	p	p
$p\,\mathbf{R}\,q$	q	q	q, p	-	-
	q	q	q	q	q

Auswertung einer LTL Formel

Beispiel



Beispiel (Auswertungen)

- $s_0 \vDash p \wedge q$
- $s_0 \vDash \neg r$
- $s_0 \vDash \mathbf{X}\, r$
- $s_0 \nvDash \mathbf{X}(q \wedge r)$
- $s_0 \vDash \mathbf{G} \neg (p \wedge r)$
- $s_2 \vDash \mathbf{G}\, r$
- Für jeden Zustand s gilt:
  $s \vDash \mathbf{F}(\neg q \wedge r) \rightarrow \mathbf{F}\,\mathbf{G}\, r$
- $s_0 \vDash \mathbf{G}\,\mathbf{F}\, p \rightarrow \mathbf{G}\,\mathbf{F}\, r$

Was geht mit LTL? Was geht nicht?

Was geht mit LTL?

- Man kann keinen Zustand erreichen, in dem `started` gilt aber nicht `ready`: $\mathbf{G}\,\neg(\texttt{started} \wedge \neg\texttt{ready})$
- Wenn in einem beliebigen Zustand ein Request eintrifft, wird er auf jeden Fall in einem zukünftigen Zustand bestätigt: $\mathbf{G}(\texttt{req} \rightarrow \mathbf{F}\,\texttt{ack})$
- Wenn ein Prozess unendlich oft aktiviert wird, läuft er unendlich oft: $\mathbf{G}\,\mathbf{F}\,\texttt{enabled} \rightarrow \mathbf{G}\,\mathbf{F}\,\texttt{running}$

Was geht nicht mit LTL?

- Von jedem Zustand aus ist es möglich in einen `restart` Zustand zu gelangen (d.h. *es gibt einen Pfad* von jedem Zustand aus zu einem Zustand, an dem `restart` gilt)
- Der Lift kann untätig mit geschlossenen Türen im dritten Stock stehen bleiben (d.h. vom Zustand in dem der Lift im dritten Stock ist, gibt es einen Pfad, in dem er dort bleibt)

Wichtige Äquivalenzen

Definition (Äquivalenz)

Zwei LTL Formeln φ und ψ sind semantisch äquivalent $\varphi \equiv \psi$, wenn für alle Modelle $\mathcal{M}$ und alle Pfade π in $\mathcal{M}$ gilt: $\pi \vDash \varphi$ gdw. $\pi \vDash \psi$.

a)	$\neg \mathbf{G}\, \varphi \equiv \mathbf{F} \neg \varphi$	Dualität von $\mathbf{G}$ und $\mathbf{F}$
b)	$\neg \mathbf{F}\, \varphi \equiv \mathbf{G} \neg \varphi$	Dualität von $\mathbf{G}$ und $\mathbf{F}$
c)	$\neg \mathbf{X}\, \varphi \equiv \mathbf{X} \neg \varphi$	Dualität von $\mathbf{X}$
d)	$\neg(\varphi_1 \,\mathbf{U}\, \varphi_2) \equiv \neg\varphi_1 \,\mathbf{R}\, \neg\varphi_2$	Dualität von $\mathbf{R}$ und $\mathbf{U}$
e)	$\neg(\varphi_1 \,\mathbf{R}\, \varphi_2) \equiv \neg\varphi_1 \,\mathbf{U}\, \neg\varphi_2$	Dualität von $\mathbf{R}$ und $\mathbf{U}$
f)	$\mathbf{G}(\varphi_1 \wedge \psi) \equiv \mathbf{G}\, \varphi_1 \wedge \mathbf{G}\, \varphi_2$	1. Distributivgesetz
g)	$\mathbf{F}(\varphi_1 \vee \psi) \equiv \mathbf{F}\, \varphi_1 \vee \mathbf{F}\, \varphi_2$	2. Distributivgesetz

Branching Time Logic

LTL

- LTL Formeln werden auf Pfaden ausgewertet
- Eine Formel gilt an einem Zustand, wenn sie an allen möglichen Pfaden von diesem Zustand aus gilt
- d.h. implizite Allquantifizierung der Pfade
- Eigenschaften, die die Existenz eines Pfades fordern, können nicht in LTL ausgedrückt werden

Branching Time Logic

- Quantifizierung über Pfade ist erlaubt
- 2 bekannte Formen:
  - **CTL:** Jedem Temporaloperator geht genau ein Pfadquantor voraus
  - **CTL*:** Quantifizierung beliebiger Formeln

$$\mathbf{CTL} \subset \mathbf{CTL^*},\ \mathbf{LTL} \subset \mathbf{CTL^*},\ \mathbf{CTL} \not\subseteq \mathbf{LTL},\ \mathbf{LTL} \not\subseteq \mathbf{CTL}$$

CTL—Syntax

EBNF-Grammatik für CTL Formeln

Formel	=	$\top \mid \bot$	Konstanten
	$\mid$	*Variable*	$\in \mathcal{V}$
	$\mid$	$\neg$ Formel	Negation
	$\mid$	Formel $\wedge$ Formel	Konjunktion
	$\mid$	Formel $\vee$ Formel	Disjunktion
	$\mid$	Formel $\rightarrow$ Formel	Implikation
	$\mid$	$\mathbf{AX}$ Formel $\mid$ $\mathbf{EX}$ Formel	Next
	$\mid$	$\mathbf{AF}$ Formel $\mid$ $\mathbf{EF}$ Formel	Finally
	$\mid$	$\mathbf{AG}$ Formel $\mid$ $\mathbf{EG}$ Formel	Globally
	$\mid$	$\mathbf{A}$ [Formel $\mathbf{U}$ Formel]	Until
	$\mid$	$\mathbf{E}$ [Formel $\mathbf{U}$ Formel]	Until
	$\mid$	(Formel)	Klammerung

- $\mathbf{A}$: Auf allen Pfaden
- $\mathbf{E}$: Es existiert ein Pfad

Auswertung einer CTL Formel

Algorithmus: $\texttt{holds}(\mathcal{M}, s, \psi)$

Eingabe: Modell $\mathcal{M} = (S, \longrightarrow, L)$, Zustand $s \in S$, CTL Formel ψ
Ausgabe: Evaluation von ψ am Zustand s

$\texttt{holds}(\mathcal{M}, s, \psi) = \psi \texttt{ match}$

$\top \rightsquigarrow \texttt{true}$

$\mid \bot \rightsquigarrow \texttt{false}$

$\mid v \in \mathcal{V} \rightsquigarrow \texttt{if } v \in L(s) \texttt{ then true else false}$

$\mid \neg\varphi \rightsquigarrow \texttt{if holds}(\mathcal{M}, s, \varphi) \texttt{ then false else true}$

$\mid \varphi_1 \wedge \varphi_2 \rightsquigarrow \texttt{if holds}(\mathcal{M}, s, \varphi_1) \texttt{ and holds}(\mathcal{M}, s, \varphi_2) \texttt{ then true else false}$

$\mid \varphi_1 \vee \varphi_2 \rightsquigarrow \texttt{holds}(\mathcal{M}, s, \neg(\neg\varphi_1 \wedge \neg\varphi_2))$

$\mid \varphi_1 \rightarrow \varphi_2 \rightsquigarrow \texttt{holds}(\mathcal{M}, s, \neg\varphi_1 \vee \varphi_2))$

$\mid \mathbf{AX}\,\varphi \rightsquigarrow \texttt{for all } s' \in S\texttt{: } s \longrightarrow s' \rightarrow \texttt{holds}(\mathcal{M}, s', \varphi)$

$\mid \mathbf{EX}\,\varphi \rightsquigarrow \texttt{exists } s' \in S\texttt{: } s \longrightarrow s' \wedge \texttt{holds}(\mathcal{M}, s', \varphi)$

$\mid \ldots$

Auswertung einer CTL Formel

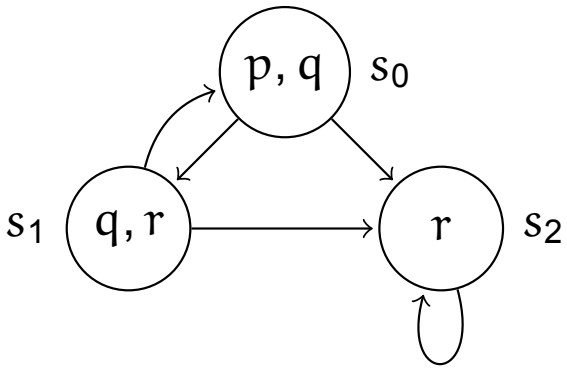
Algorithmus: $\texttt{holds}(\mathcal{M}, s, \psi)$

	...
	$\mathbf{AG}\,\varphi \rightsquigarrow$ for all $s \longrightarrow s_1 \longrightarrow \ldots$ for all s_i: $\texttt{holds}(\mathcal{M}, s_i, \varphi)$
	$\mathbf{EG}\,\varphi \rightsquigarrow$ exists $s \longrightarrow s_1 \longrightarrow \ldots$ for all s_i: $\texttt{holds}(\mathcal{M}, s_i, \varphi)$
	$\mathbf{AF}\,\varphi \rightsquigarrow$ for all $s \longrightarrow s_1 \longrightarrow \ldots$ exists s_i: $\texttt{holds}(\mathcal{M}, s_i, \varphi)$
	$\mathbf{EF}\,\varphi \rightsquigarrow$ exists $s \longrightarrow s_1 \longrightarrow \ldots$ exists s_i: $\texttt{holds}(\mathcal{M}, s_i, \varphi)$
	$\mathbf{A}[\varphi_1\,\mathbf{U}\,\varphi_2] \rightsquigarrow$ for all $\pi = s \longrightarrow s_1 \longrightarrow \ldots$: $\texttt{holds}(\mathcal{M}, \pi, \varphi_1\,\mathbf{U}\,\varphi_2)$
	$\mathbf{E}[\varphi_1\,\mathbf{U}\,\varphi_2] \rightsquigarrow$ exists $\pi = s \longrightarrow s_1 \longrightarrow \ldots$: $\texttt{holds}(\mathcal{M}, \pi, \varphi_1\,\mathbf{U}\,\varphi_2)$

- $\mathbf{E}$ und $\mathbf{A}$ quantifizieren immer über Pfade
- $\mathbf{G}$ und $\mathbf{F}$ quantifizieren über Zustände
- $\mathbf{E}$ und $\mathbf{A}$ sind dual zueinander
- Gilt $\texttt{holds}(\mathcal{M}, s, \varphi)$, so schreiben wir auch $s \vDash_{\mathcal{M}} \varphi$ oder $s \vDash \varphi$

Auswertung einer CTL Formel

Beispiel



Beispiel (Auswertungen)

- $s_0 \models \mathbf{EX}(q \wedge r)$
- $s_0 \models \neg\, \mathbf{AX}(q \wedge r)$
- $s_0 \not\models \mathbf{EF}(p \wedge r)$
- $s_2 \models \mathbf{EG}\, r$
- $s_0 \models \mathbf{AF}\, r$
- $s_0 \models \mathbf{E}[(p \wedge q)\, \mathbf{U}\, r]$
- $s_0 \models \mathbf{A}[p\, \mathbf{U}\, r]$
- $s_0 \models \mathbf{AG}(p \vee q \vee r \rightarrow \mathbf{EF}\, \mathbf{EG}\, r)$

Was kann CTL?

👍 Klassische Fragestellungen in CTL

- Gibt es einen Zustand, in dem `started` gilt, jedoch `ready` nicht: $\mathbf{EF}(\texttt{started} \wedge \neg \texttt{ready})$
- Wird in einem beliebigen Zustand ein Request empfangen, wird er auch irgendwann bestätigt: $\mathbf{AG}(\texttt{req} \rightarrow \mathbf{AF}\,\texttt{ack})$
- Was auch immer passiert, irgendwann wird ein bestimmter Prozess permanent im Deadlock sein: $\mathbf{AF}\,\mathbf{AG}\,\texttt{deadlock}$
- Aus jeden Zustand ist es möglich zu einen Restart Zustand zu kommen: $\mathbf{AG}\,\mathbf{EF}\,\texttt{restart}$
- Der Lift kann untätig mit geschlossenen Türen im dritten Stock stehen bleiben: $\mathbf{AG}(\texttt{floor3} \wedge \texttt{idle} \wedge \texttt{doorclosed} \rightarrow \mathbf{EG}(\texttt{floor3} \wedge \texttt{idle} \wedge \texttt{doorclosed}))$

? Fragestellung des Modelcheckings

Gegeben ein Modell, in welchen Zuständen gilt eine bestimmte Formel in CTL oder LTL?

Literaturtipps

Literaturhinweis

- *M. Huth & M. Ryan.* **Logic in Computer Science** **Chapter 3.** Cambridge University Press, 2004.