2.3 Baum-Zeit Logik

Zur Vorlesung

Embedded Systems

WS 14/15

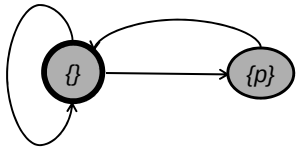
Reiner Kolla

Baum-Zeit Logik

Linear-Zeit Logik ist eine Logik, die sich auf alle unendlichen Sequenzen von Belegungen mit atomaren Eigenschaften bezieht. Es gibt nur eine mögliche Zukunft. Man kann nicht über alternative Möglichkeiten den Entwicklung Aussagen machen.

Baum-Zeit Logik erlaubt Aussagen über die Möglichkeiten eines Systems

Prominentester Vertreter ist **CTL**. (Computation Tree Logic)



Beispiel:
Nebenstehendes Verhalten lässt sich nicht in LTL adäquat beschreiben. Es gibt sowohl Folgen in denen p nie gilt als auch Folgen in denen p irgendwann mal gilt.

Belegungsbaum zu einer Kripke Struktur

Sei $K = (S, \rightarrow, s_0, AP, L)$ eine Kripke Struktur mit Startzustand s_0.

Für $s \in S$ sei ein Belegungsbaum $T_K(s)$ zu K mit Wurzel s wie folgt definiert:

Zu jedem $s' \in S$ mit $s \rightarrow s'$ gibt es genau einen zu $T_K(s')$ isomorphen Unterbaum von $T_K(s)$.

$T_K(s)$ beschreibt quasi eine Entfaltung der Kripke Struktur in einen unendlich tiefen Baum.

CTL Formeln werden über Berechnungsbäume interpretiert. Sie machen bei jedem temporalen Operator eine Aussage über die Menge der möglichen Entwicklungen im Baum durch existenzielle oder universelle Quantifizierung.

Syntax von CTL

Sei *AP* eine Menge von atomaren Aussagen. Dann ist die Menge der CTL Formeln über *AP* wie folgt definiert:

1. Jedes $p \in AP$ ist eine CTL Formel.
2. Sind ϕ_1 und ϕ_2 Formeln, dann auch
   $\neg\phi_1$, $\phi_1 \vee \phi_2$, $E\boldsymbol{X}\phi_1$, $E\boldsymbol{G}\phi_1$, $\phi_1\ E\boldsymbol{U}\phi_2$
3. Nichts sonst ist eine CTL Formel.

Dies ist wieder eine sehr minimalistische Definition. ***EX*** (exists next), ***EG*** (exists globally), und ***EU*** (exists until) sind temporale Operatoren.

Alle andere Operatoren kann man durch Formeln über diesen ausdrücken. Für Beweise ist diese Definition nützlich, weil man nicht so viele Fälle unterscheiden muss.

Nützliche „Abkürzungen“ für CTL Formeln:

Wir erweitern CTL um ein paar Notationen, deren Bedeutung wir aber auf unsere Minimal-CTL zurückführen. Wenn man Eigenschaften formulieren will, kann man den Komfort nutzen, in Beweisen zu allgemeinen Eigenschaften der CTL, muss man dazu nichts zeigen:

$\phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2)$ $\quad\quad$ $\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$

$true \equiv a \vee \neg\, a$ $\quad\quad$ $false \equiv \neg\, true$

$\mathbf{EF}\phi \equiv true\ \mathbf{EU}\phi$ $\quad\quad$ $\phi_1 \mathbf{EW}\ \phi_2 \equiv (\phi_1 \mathbf{EU}\ \phi_2) \vee \mathbf{EG}\ \phi_1$

$\mathbf{AX}\phi \equiv \neg\mathbf{EX}\neg\phi$ $\quad\quad$ $\mathbf{AG}\phi \equiv \neg\mathbf{EF}\neg\phi$

$\mathbf{AF}\phi \equiv \neg\mathbf{EG}\neg\phi$ $\quad\quad$ $\phi_1 \mathbf{AW}\ \phi_2 \equiv \neg(\neg\phi_2 \mathbf{EU} \neg\ (\phi_1 \vee \phi_2))$

$\phi_1 \mathbf{AU}\ \phi_2 \equiv \mathbf{AF}\phi_2 \wedge (\phi_1 \mathbf{AW}\ \phi_2)$

Dabei steht

A für „All“ (für alle Pfade), **F** für „Future“

W für „weak until“

Erfüllung einer CTL Formel

Sei *K = (S,→, s_0, AP,L)* eine Kripke Struktur mit Startzustand s_0. Und sei

$T_K(s)$ ein Belegungsbaum beginnend beim Zustand *s*.

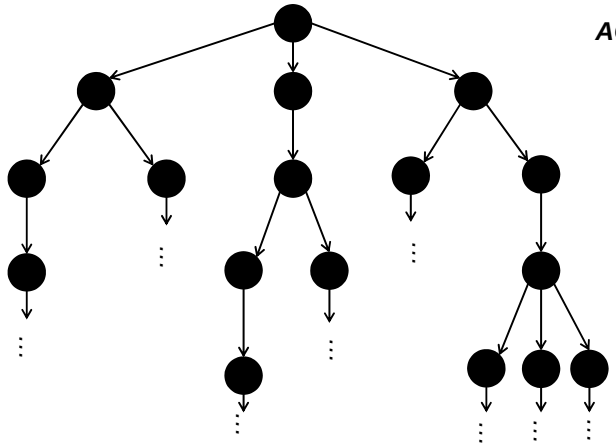
Wir sagen $T_K(s)$ **erfüllt eine CTL Formel** ϕ, $T_K(s) \vDash \phi$ nach folgender Fallunterscheidung:

- Für $p \in AP$: $T_K(s) \vDash p$ genau dann wenn $p \in L(s)$
- $T_K(s) \vDash \neg\phi$ genau, dann wenn $T_K(s) \nvDash \phi$
- $T_K(s) \vDash \phi_1 \vee \phi_2$ genau dann, wenn $T_K(s) \vDash \phi_1$ oder $T_K(s) \vDash \phi_2$
- $T_K(s) \vDash \mathrm{E}\boldsymbol{X}\phi$ genau dann, wenn *s* einen Nachfolger *s‘* im Baum hat und $T_K(s') \vDash \phi$
- $T_K(s) \vDash \mathrm{E}\boldsymbol{G}\phi$ genau dann, wenn es einen unendlichen Pfad $s = s_1 \rightarrow s_2 \rightarrow s_2 \cdots$ gibt mit $\forall i$: $T_K(s_i) \vDash \phi$
- $T_K(s) \vDash \phi_1\ \mathrm{E}\boldsymbol{U}\phi_2$ genau dann wenn es einen unendlichen Pfad $s = s_1 \rightarrow s_2 \rightarrow s_2 \cdots$ gibt mit $\exists k$: $(T_K(s_k) \vDash \phi_2$ und $\forall i < k$: $T_K(s_k) \vDash \phi_1)$

Beispiele

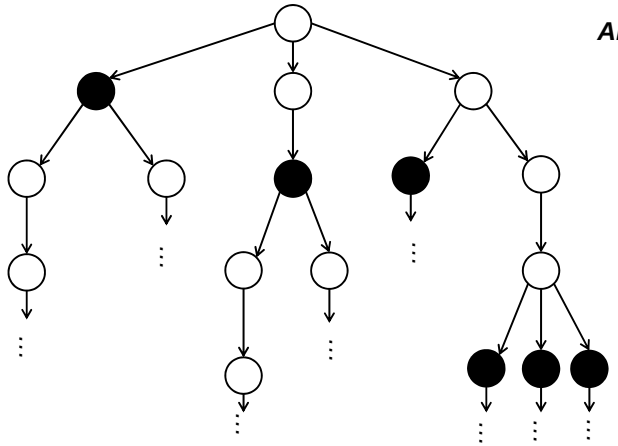
Sei $\mathrm{AP} = \{\mathrm{rot}, \mathrm{schwarz}\}$ und die roten Knoten im Baum haben Eigenschaft rot, die schwarzen die Eigenschaft schwarz. Weiß stehe für beliebig. Folgende Bäume erfüllen die nebenstehende Formel

***AG** schwarz*

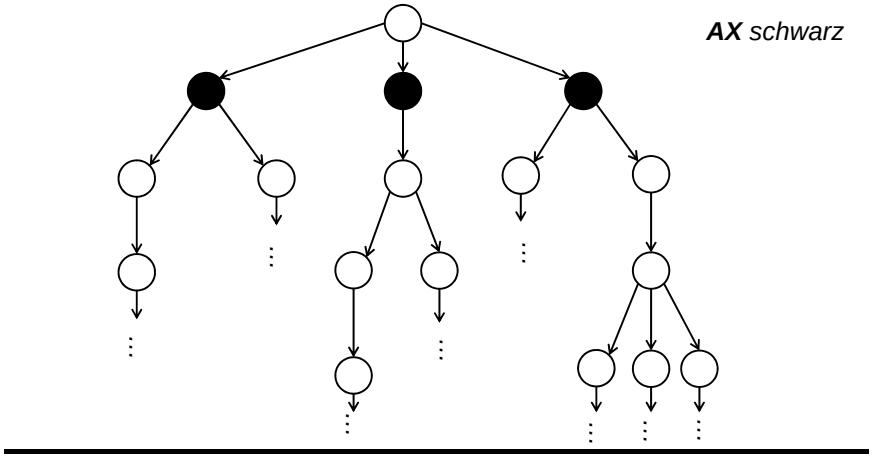


Beispiele

***AF** schwarz*

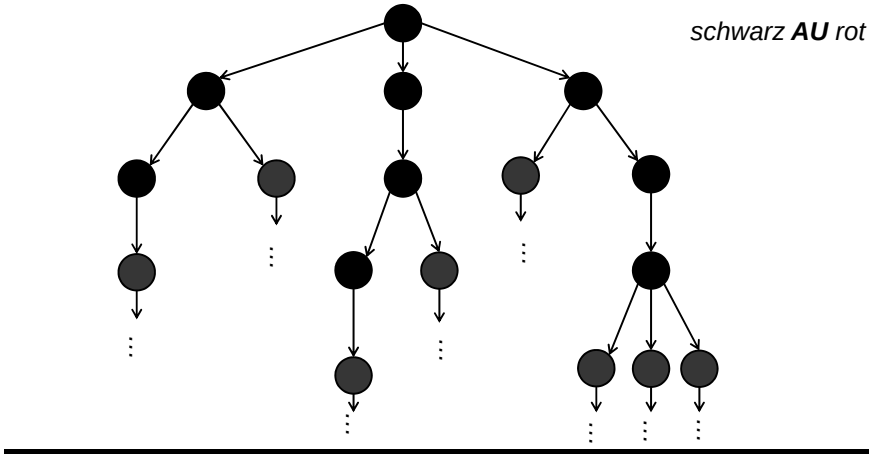


Beispiele



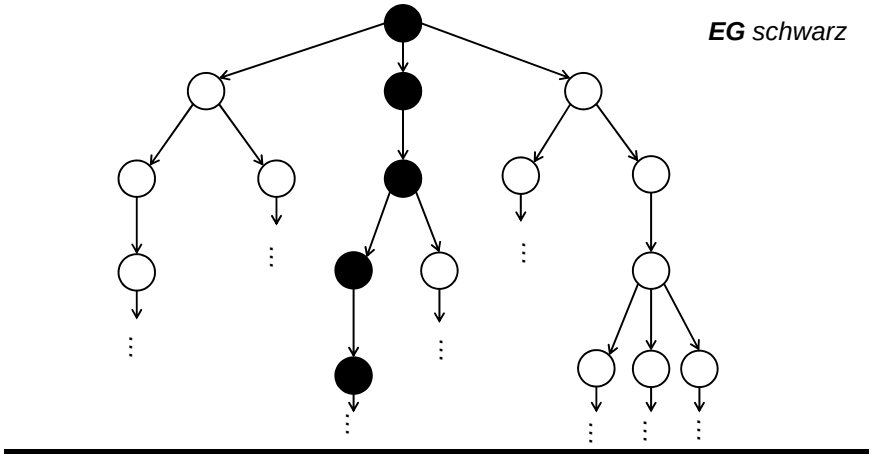
***AX** schwarz*

Beispiele



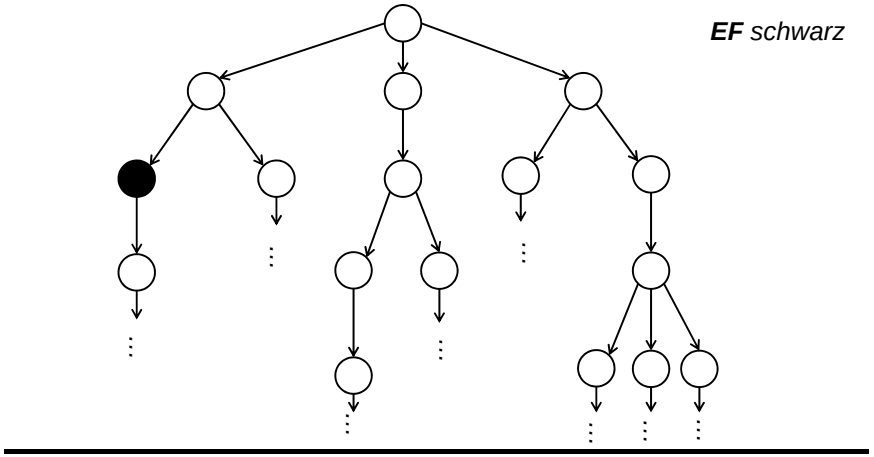
*schwarz **AU** rot*

Beispiele



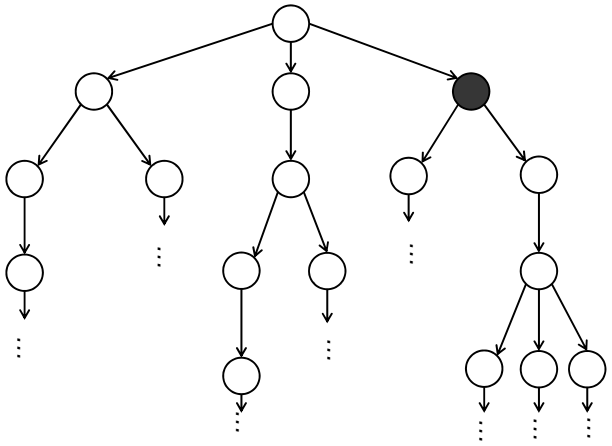
***EG** schwarz*

Beispiele



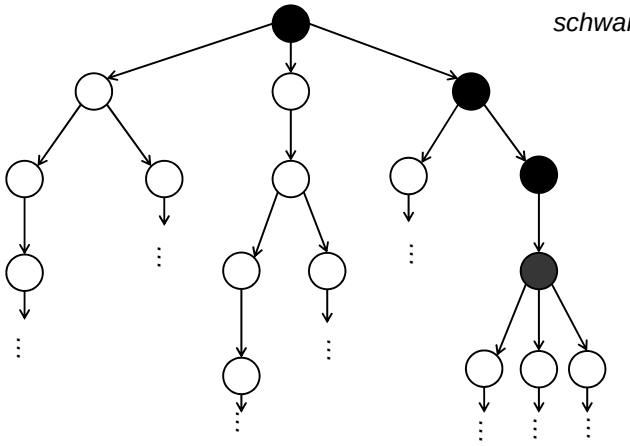
***EF** schwarz*

Beispiele



***EX** rot*

Beispiele



*schwarz **EU** rot*

Das CTL Model Checking Problem

Wir wollen nun anschauen, wie man (im Prinzip, schnelle Heuristiken sind nach wie vor Gegenstand der aktuellen Forschung) Model Checking für CTL Formeln algorithmisch entscheidet:

Problem: -- CTL Model Checking

Gegeben sei eine Kripke Struktur *K= (S, →, s_0, AP, L)* und eine CTL Formel ϕ über *AP* . Entscheide, ob $T_K(s_0) \vDash \phi$?

Wenn $T_K(s_0) \vDash \phi$ sagt man auch: *K* ist ein Model für ϕ

Wir definieren ferner $[\![\phi]\!]_K := \{\boldsymbol{s} \in \boldsymbol{S} | T_K(s) \vDash \phi\}$

Damit ist *K* Modell für ϕ genau dann wenn $s_0 \in [\![\phi]\!]_K$

Problem: -- CTL global Model Checking

Gegeben sei eine Kripke Struktur *K= (S, →, s_0, AP, L)* und eine CTL Formel ϕ. Berechne $[\![\phi]\!]_K$

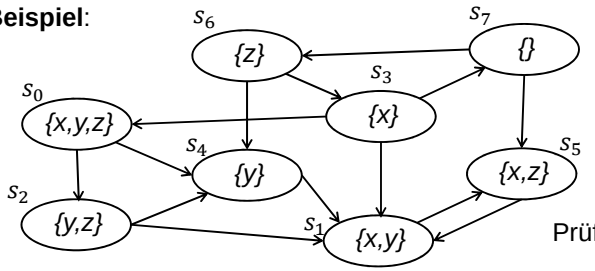
Lösen des global CTL Model Checking Problems

$T_K(s_0)$ besteht stets nur aus unendlichen Unterbäumen $T_K(s)$ für $s \in S$. Wenn man Model Checking durchführen will, genügt es also wenn man für jeden Zustand *s* und jede Teilformel ϕ′ von ϕ beginnend mit den einfachsten Teilformeln entscheidet ob $T_K(s) \vDash \phi$‘.

(Dynamisches Programmieren)

Idee: Berechne für alle Teilformeln ϕ′ mit aufsteigender Größe die Menge $[\![\phi']\!]_K$. Prüfe am Ende ob $s_0 \in [\![\phi]\!]_K$.

Beispiel:

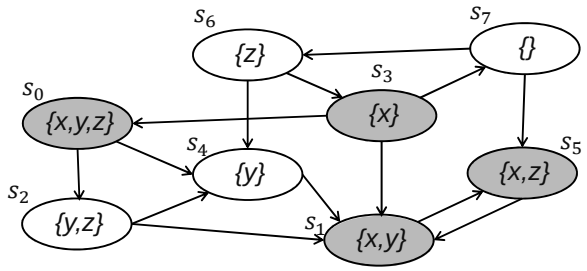


Prüfe, ob ***AF AG** x* gilt.

Global CTL Model Checking am Beispiel

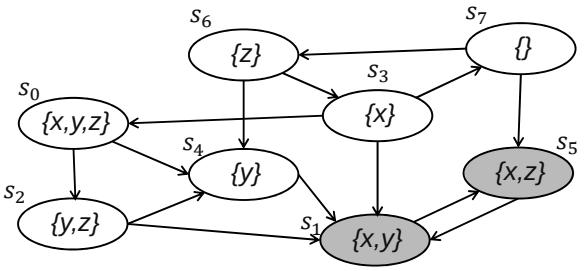
Berechne nacheinander die Menge $[\![x]\!]_K$, $[\![\boldsymbol{AG}x]\!]_K$, $[\![\boldsymbol{AFAG}x]\!]_K$

$[\![x]\!]_K$:



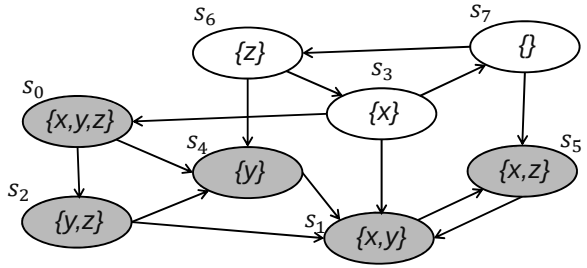
Global CTL Model Checking am Beispiel

$[\![\boldsymbol{AG}x]\!]_K$:



Global CTL Model Checking am Beispiel

$[\![\boldsymbol{AFAG}x]\!]_K$:



Beispiel: Speisende Philosophen als CTL

Elementare Aussagen:

$isst(i)$ für *i= 1,…,5*

Aussagen:

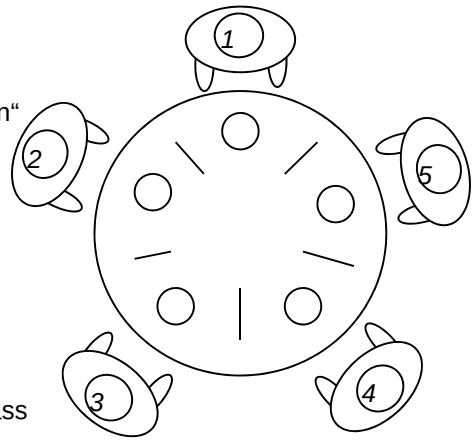
„Niemals kann 1,2 gleichzeitig essen“

$AG\neg(isst(1) \wedge isst(2))$

analog für 2,3 3,4 4,5 und 5,1

„möglicherweise isst 1 nie“

$EG\ \neg isst(1)$



„Es ist in jeder Situation möglich, dass irgendwann nur Philosoph 2 isst.“

$AG\ EF\neg isst(1) \wedge isst(2) \wedge \neg isst(3) \wedge \neg isst(4) \wedge \neg isst(5)$

Lösen des global CTL Model Checking Problems

Erinnerung: Berechne für alle Teilformeln ϕ' mit aufsteigender Größe die Menge $[\![\phi']\!]_K$. Prüfe am Ende ob $s_0 \in [\![\phi]\!]_K$.

Die bottom up Methode

1. Für alle $p \in AP$ setzen wir $\mu(p) := [\![p]\!]_K = \{s|L(s) = p\}$.
2. Sei ψ eine einfache Teilformel der Form $\neg p_1$, $p_1 \vee p_2$, $E\boldsymbol{X}p_1$, $E\boldsymbol{G}p_1$, p_1 $E\boldsymbol{U}p_2$wobei die $p_i \in AP$ dann berechnen wir die Menge $[\![\psi]\!]_K$ nach den im den folgenden Folien beschriebenen Methoden.
3. Erweitere *AP* um eine neue atomare Aussagen *q*, d.h wir setzen $AP = AP \cup \{q\}$, setze $\mu(q) := [\![\psi]\!]_K$aus Schritt 2 und ersetze alle Vorkommen der Teilformel ψ in ϕ durch *q*. Gehe zu Schritt 2 solange nicht $\phi = q$ gilt.

Berechnung von $[\![\neg p_1]\!]$, $[\![p_1 \vee p_2]\!]$, $[\![E\boldsymbol{X}p_1]\!]$

1. Ist $\psi = \neg p_1$, dann ist nach Definition $[\![\psi]\!]_K = S \setminus \mu(p_1)$

2. Ist $\psi = p_1 \vee p_2$ dann ist $[\![\psi]\!]_K = \mu(p_1) \cup \mu(p_2)$

3. Ist $\psi = E\boldsymbol{X}p_1$ dann setzen wir
   $[\![\psi]\!]_K = \{s|\exists t \in \mu(p_1)\ mit\ s \to t\} =: Pre(\mu(p_1))$

Wir werden nun noch die Menge $[\![E\boldsymbol{G}p_1]\!]_K$, als größten Fixpunkt einer Gleichung

und die Menge $[\![p_1\ E\boldsymbol{U}p_2]\!]_K$ als kleinsten Fixpunkt einer Gleichung definieren.

Berechnung von $[\![E\boldsymbol{G}p]\!]$

Wir zeigen zunächst

Lemma 1:

$[\![E\boldsymbol{G}p]\!]_K$ ist größte (bzgl $\subseteq$) Lösung der Gleichung

$$X = \mu(p) \cap Pre(X)$$

Wobei $Pre(X) := \{s|\exists t \in X\ mit\ s \to t\}$ die Vorgängermenge bezüglich der Transitionsrelation ist.

Beweis:

Wir zeigen zunächst, dass $[\![E\boldsymbol{G}p]\!]_K$Lösung ist.

Erinnerung

$[\![E\boldsymbol{G}p]\!]_K = \{s|$ ∃Pfad $s = s_1 \to s_2 \to s_3 \cdots$ mit $\forall i: L(s_i) = p\}$

Berechnung von $[\![E\boldsymbol{G}p]\!]$ -- ff

$\Rightarrow$: Sei $s \in [\![E\boldsymbol{G}p]\!]_K$.

Dann ist $L(s) = p$ und damit $s \in \mu(p)$.

Ferner ist mit $s \in [\![E\boldsymbol{G}p]\!]_K$ insbesondere auch $s_2 \in [\![E\boldsymbol{G}p]\!]_K$.

Also gilt wegen $s = s_1 \to s_2$ schon $s \in Pre([\![E\boldsymbol{G}p]\!]_K)$ und damit gilt $s \in \mu(p) \cap Pre([\![E\boldsymbol{G}p]\!]_K)$

Also gilt $[\![E\boldsymbol{G}p]\!] \subseteq \mu(p) \cap Pre([\![E\boldsymbol{G}p]\!]_K)$

$\Leftarrow$: Sei $s \in \mu(p) \cap Pre([\![E\boldsymbol{G}p]\!]_K)$

Da $s \in Pre([\![E\boldsymbol{G}p]\!]_K)$ gibt es einen Nachfolger *t* von *s* für den es einen unendlichen Pfad $t = t_1 \to t_2 \to t_3 \cdots$ mit $\forall i: L(t_i) = p$ gibt.

Da aber auch $s \in \mu(p)$ ist auch $L(s) = p$ damit ist $s = s_1 \to s_2 \to s_3 \cdots$ mit $s_i = t_{i-1}$ für *i>1* ein unendlicher Pfad mit $\forall i: L(s_i) = p$

Also ist damit $s \in [\![E\boldsymbol{G}p]\!]_K$ und damit gilt

$[\![E\boldsymbol{G}p]\!] \supseteq \mu(p) \cap Pre([\![E\boldsymbol{G}p]\!]_K)$

Berechnung von $[\![\mathrm{E}\boldsymbol{G}p]\!]$ -- ff

Also gilt „=“ und $[\![\mathrm{E}\boldsymbol{G}p]\!]$ ist Lösung der Fixpunktgleichung.

Wir müssen nur noch zeigen, dass es keine Obermenge von $[\![\mathrm{E}\boldsymbol{G}p]\!]$ gibt, die Fixpunkt dieser Gleichung ist. Sei also *M* ein beliebiger Fixpunkt von $\mu(p) \cap Pre(M)$

Wir zeigen $M \subseteq [\![\mathrm{E}\boldsymbol{G}p]\!]$:

Sei also $s \in M$ beliebig und demnach $s \in \mu(p)$ und $s \in Pre(M)$.

Dann gibt es wegen $s \in Pre(M)$ ein $s_2 \in M$ mit $s = s_1 \to s_2$ und es ist $\forall i = 1{,}2: L(s_i) = p$ da ja $s_1 = s \in \mu(p)$

Nehmen wir an, wir hätten eine Folge $s = s_1 \to s_2 \to \ldots \to s_k$konstruiert mit $\forall i \leq k: L(s_i) = p$ und $s_i \in M$

Dann gibt es weil mit der Fixpunktgleichung insbesondere

$s_k \in M \subseteq Pre(M)$ ein $s_{k+1} \in M$ mit $s_k \to s_{k+1}$

Also können wir induktiv eine unendliche Folge $s = s_1 \to s_2 \to \ldots$ konstruieren mit $\forall i = 1{,}2: L(s_i) = p$ damit ist aber $s \in [\![\mathrm{E}\boldsymbol{G}p]\!]$ also gilt $M \subseteq [\![\mathrm{E}\boldsymbol{G}p]\!]$

Berechnung von $[\![\mathrm{E}\boldsymbol{G}p]\!]$ -- ff

Wir wollen nun zeigen, dass wir diesen größten Fixpunkt iterativ berechnen können:

Lemma 2:

Sei *f* als Funktion auf Zustandsmengen wie folgt definiert

$$\mathrm{f}(X) := \mu(p) \cap Pre(X)$$

Dann gilt

1. *f* ist monoton, d.h. $X \supseteq X' \Longrightarrow f(X) \supseteq f(X')$
2. Sei $f^{i+1}(X) := f(f^i(X))$ dann gilt $\forall i > 0: f^i(S) \supseteq [\![\mathrm{E}\boldsymbol{G}p]\!]_K$

Beweis:

Zu 1.:

$X \supseteq X' \Longrightarrow Pre(X) \supseteq Pre(X') \Longrightarrow \mu(p) \cap Pre(X) \supseteq \mu(p) \cap Pre(X')$

$\Leftrightarrow f(X) \supseteq f(X')$

Berechnung von $[\![\mathrm{E}\boldsymbol{G}p]\!]$ -- ff

Zu 2.:

Es gilt offensichtlich $[\![\mathrm{E}\boldsymbol{G}p]\!]_K \subseteq S =: f^0(S)$

Induktiv folgt dann auch

$f^{i+1}(X) := \mu(p) \cap Pre(f^i(X)) \supseteq \mu(p) \cap Pre([\![\mathrm{E}\boldsymbol{G}p]\!]_K)$ mit I.A.

$= [\![\mathrm{E}\boldsymbol{G}p]\!]_K$ mit Lemma 1

q.e.d

Wir haben also eine monoton fallende Folge

$S \supseteq f(S) \supseteq f^2(S) \supseteq f^3(S) \supseteq f^4(S) \supseteq \cdots$

Für diese gilt folgendes

Berechnung von $[\![\mathrm{E}\boldsymbol{G}p]\!]$ -- ff

Lemma 3:

Es gibt einen Index *i* mit $f^{i+1}(S) = f^i(S)$

und es gilt dann $f^i(S) = [\![\mathrm{E}\boldsymbol{G}p]\!]_K$

Beweis:

Da S endlich ist und die Folge monoton fällt, endet nach sie endlich vielen Schritten bei einem Fixpunkt.

Für jedes *j* ist $f^j(S) \supseteq [\![\mathrm{E}\boldsymbol{G}p]\!]_K$

Also gilt auch für den Fixpunkt $f^i(S) \supseteq [\![\mathrm{E}\boldsymbol{G}p]\!]_K$

Da $[\![\mathrm{E}\boldsymbol{G}p]\!]_K$ nach Lemma 1 größter Fixpunkt von *f* ist, gilt aber auch

$f^i(S) \subseteq [\![\mathrm{E}\boldsymbol{G}p]\!]_K$, da $f^i(S)$ Fixpunkt von f ist, also gilt = q.e.d.

Damit können wir $[\![\mathrm{E}\boldsymbol{G}p]\!]_K$ durch eine Fixpunktiteration berechnen.

Berechnung von $[\![p_1 \ \mathrm{E}\boldsymbol{U} p_2]\!]$

Analog bzw dual zu den Beweisen von Lemma 1 bis 3 überlegt man sich

Lemma 4:

$[\![p_1 \ \mathrm{E}\boldsymbol{U} p_2]\!]_K$ ist kleinste (bzgl ⊆) Lösung der Gleichung

$$X = \mu(p_2) \cup (\mu(p_1) \cap Pre(X))$$

Lemma 5:

Sei *F* als Funktion auf Zustandsmengen wie folgt definiert

$$\mathrm{F}(X) := \mu(p_2) \cup \big(\mu(p_1) \cap Pre(X)\big)$$

Dann gilt

1. *F* ist monoton, d.h. $X \supseteq X' \Longrightarrow f(X) \supseteq f(X')$
2. Sei $F^{i+1}(X) := F(F^i(X))$ dann gilt $\forall i > 0: F^i(\emptyset) \subseteq [\![p_1 \ \mathrm{E}\boldsymbol{U} p_2]\!]_K$

Berechnung von $[\![p_1 \ \mathrm{E}\boldsymbol{U} p_2]\!]$ -- ff

Wir haben also diesmal eine monoton steigende Folge

$$\emptyset \subseteq F(\emptyset) \subseteq F^2(\emptyset) \subseteq F^3(\emptyset) \subseteq F^4(\emptyset) \subseteq \cdots$$

Für die dann folgendes gilt

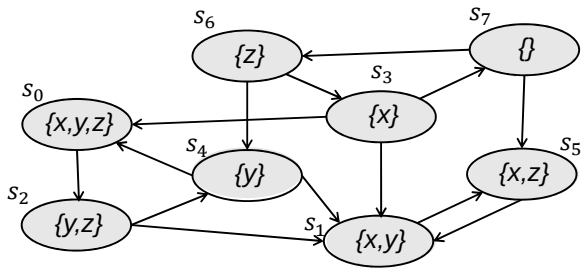
Lemma 6:

Es gibt einen Index *i* mit $F^{i+1}(\emptyset) = F^i(\emptyset)$

und es gilt dann $F^i(\emptyset) = [\![p_1 \ \mathrm{E}\boldsymbol{U} p_2]\!]_K$

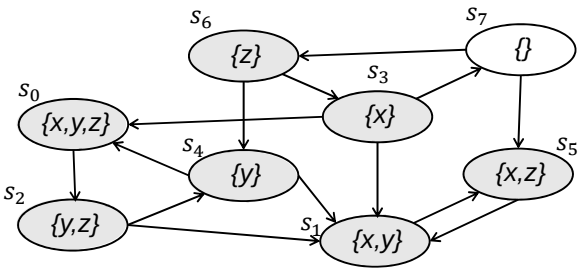
Damit können wir auch $[\![p_1 \ \mathrm{E}\boldsymbol{U} p_2]\!]_K$ durch eine Fixpunktiteration berechnen.

Beispiel: Fixpunktiteration für $[\![\mathrm{E}\boldsymbol{G} y]\!]_K$



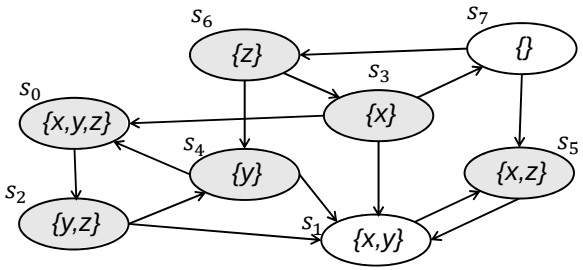
$$f^1(S) := \mu(y) \cap Pre(S) = \mu(y) \cap S$$

Beispiel: Fixpunktiteration für $[\![\mathrm{E}\boldsymbol{G} y]\!]_K$



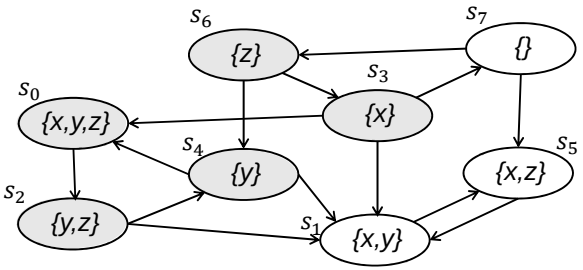
$$f^2(S) := \mu(y) \cap Pre(f^1(S))$$

Beispiel: Fixpunktiteration für $[\![\mathsf{E}\boldsymbol{G}y]\!]_K$



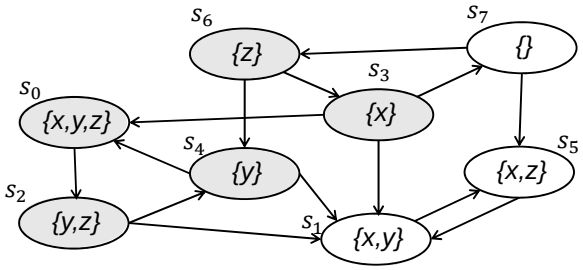
$f^2(S) \coloneqq \mu(y) \cap Pre(f^1(S))$

Beispiel: Fixpunktiteration für $[\![\mathsf{E}\boldsymbol{G}y]\!]_K$



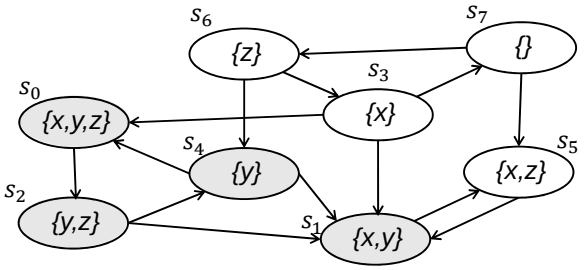
$f^3(S) \coloneqq \mu(y) \cap Pre(f^2(S))$

Beispiel: Fixpunktiteration für $[\![\mathsf{E}\boldsymbol{G}y]\!]_K$



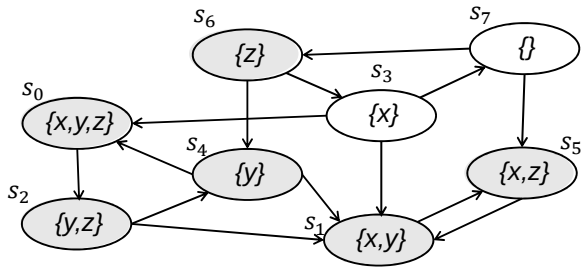
$f^3(S) \coloneqq \mu(y) \cap Pre(f^2(S)) \;\; = f^2\,(S) \;\; = [\![\boldsymbol{EG}y]\!]_K$

Beispiel: Fixpunktiteration für $[\![z\ \mathsf{E}\boldsymbol{U}y]\!]_K$



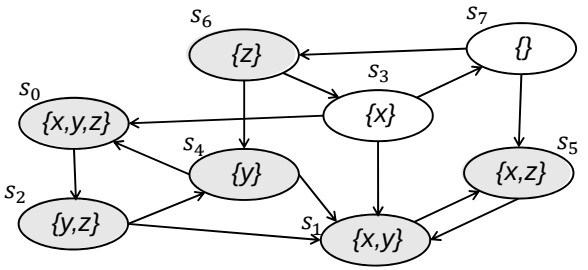
$F^1(\emptyset) \coloneqq \mu(y) \cup (\mu(z) \cap Pre(\emptyset))$

Beispiel: Fixpunktiteration für $[\![z\ \mathsf{E}\mathbf{U}y]\!]_K$



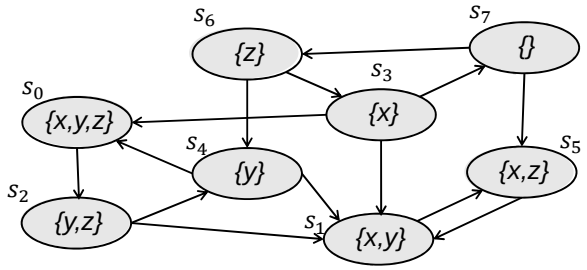
$F^2(\emptyset) \coloneqq \mu(y) \cup (\mu(z) \cap Pre(F^1(\emptyset)))$

Beispiel: Fixpunktiteration für $[\![z\ \mathsf{E}\mathbf{U}y]\!]_K$



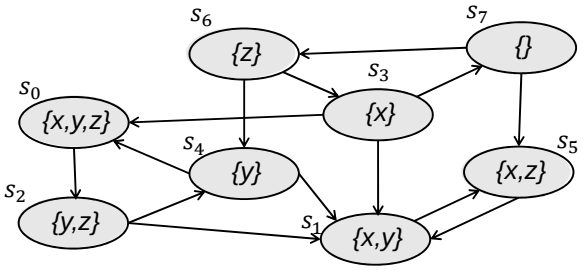
$F^2(\emptyset) \coloneqq \mu(y) \cup (\mu(z) \cap Pre(F^1(\emptyset)))$

Beispiel: Fixpunktiteration für $[\![z\ \mathsf{E}\mathbf{U}y]\!]_K$



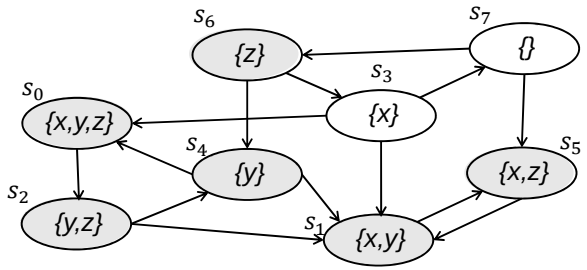
$F^3(\emptyset) \coloneqq \mu(y) \cup (\mu(z) \cap Pre(F^2(\emptyset)))$

Beispiel: Fixpunktiteration für $[\![z\ \mathsf{E}\mathbf{U}y]\!]_K$



$F^3(\emptyset) \coloneqq \mu(y) \cup (\mu(z) \cap Pre(F^2(\emptyset)))$

Beispiel: Fixpunktiteration für $[\![z\ \mathsf{E}\mathbf{U}y]\!]_K$



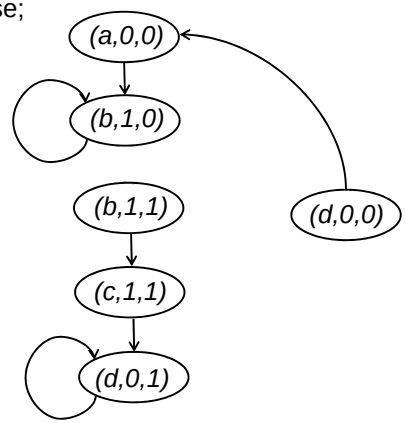
$F^3(\emptyset) := \mu(y) \cup (\mu(z) \cap Pre(F^2(\emptyset))) = F^2\ (S) = [\![z\,\mathbf{EU}y]\!]_K$

Beispiel: Handshake Algorithmus

Folgendes C Programm beschreibt ein Handshake Verfahren zwischen zweier Prozessen über zwei Flags req, ack:

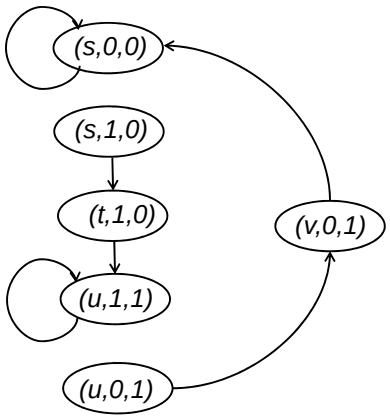
```
// globale Variablendeklaration
boolean req = false; boolean ack = false;

….
// Prozess Source
  while (true)
a: { req = true;
b:   while (!ack){ };
     //lege Frage bereit
c:   req = false;
d:   while (ack) { }  // erwarte Quittung
      // Hole Antwort ab
   }
```



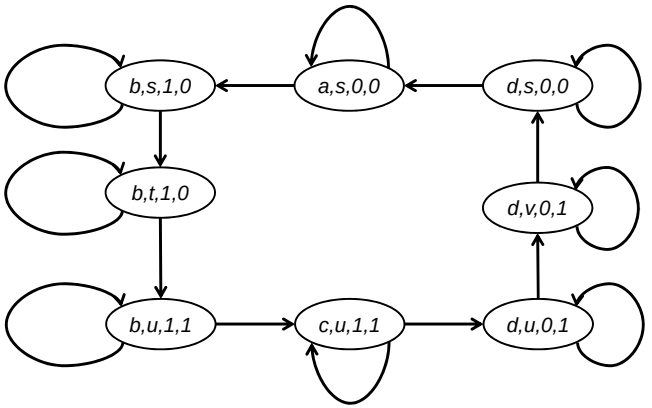
Beispiel: Handshake ff

```
….
// Prozess Drain
  while (true)
s: { while(!req) { };
t:   ack = true;
     //verarbeite Frage
u:   while (req) { }; // erwarte Quittung
v:   ack = false; // Bestätige Bearbeitung
   }
```



Beispiel: Handshake ff

Nun konstruieren wir das Transitionssystem des Gesamtsystems:

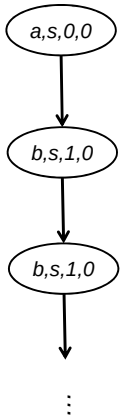


Eine interessante Eigenschaft wäre $AG(req \rightarrow reqAUack)$

ebenso $AG(\neg req \rightarrow (\neg reqAU\neg ack)$

Beispiel: Handshake ff

Folgender Pfad widerlegt $AG(req \rightarrow reqAUack)$
im Zustand (a,s,0,0)



Im Grunde widerlegt jeder unendliche Pfad die Formel, der durch einen unfairen Prozess entsteht, d.h. einer der beiden Prozesse kommt nicht dran.

CTL mit Fairness

Fairness bedeutet, dass Zustände mit bestimmten Eigenschaften unendlich of vorkommen müssen. Man kann Fairness in LTL nicht aber in CTL ausdrücken ($AGAFfair \rightarrow \phi$ macht nicht das Richtige)
Ausweg: Man definiert die temporären Operatoren auf fairen Pfaden (CTL mit Fairness). Dazu muss man nur die Operatoren betrachten, die auf unendlichen Pfaden definiert sind:

- $T_K(s) \vDash \boldsymbol{EG}_{fair}\phi$ genau dann, wenn es einen unendlichen fairen Pfad $s = s_1 \rightarrow s_2 \rightarrow s_3 \cdots$ *gibt mit* $\forall i$: $T_K(s_i) \vDash \phi$
- $T_K(s) \vDash \phi_1 \, \boldsymbol{EU}_{fair}\phi_2$ genau dann wenn es einen unendlichen fairen Pfad $s = s_1 \rightarrow s_2 \rightarrow s_3 \cdots$ *gibt mit* $\exists k$: *(*$T_K(s_k) \vDash \phi_2$ *und* $\forall i < k$: $T_K(s_k) \vDash \phi_1$*)*

Eine Fairnessbedingung können wir dann o.E. als eine Eigenschaft *p* auffassen, die auf einer Teilmenge der Zustände $\mu(p)$ gilt. Ein Pfad *s* heißt dann fair wenn $\#\{i | s(i) \in \mu(p)\}$ unendlich ist.

Faire Pfade -- Beobachtungen

Ein Pfad *s* ist fair, gdw. für jedes *i* ist der Pfad s^i fair.

Ein Pfad *s* ist fair, gdw. es für ein *i>0* einen fairen Suffix s^i gibt.

Daraus lässt sich für das Until aber ableiten:

$$\phi_1 \, EU_{fair}\phi_2 \equiv \phi_1 \, EU \, (\phi_2 \wedge EG_{fair} true)$$

D.h. wir brauchen nur eine Implementierung für den Operator EG_{fair}

Dies erhält man anschaulich, indem man alle nichttrivialen starken ZHK berechnet, die $\mu(p)$ schneiden und alle Zustände aufnimmt, die solche ZHKs erreichen können.

CTL versus LTL

Es gibt Dinge, die sich in beiden Sprachen gleichermaßen gut formulieren lassen:

Invarianten: Eigenschaften die immer gelten, z.B „p passiert nie“
in CTL: $\boldsymbol{AG}\neg p$
in LTL: $\mathbf{G}\neg p$

Reaktivität: „Wenn p passiert gilt irgendwann q“
in CTL: $\mathbf{AG}(p \rightarrow AFq)$
in LTL: $\mathbf{G}\,(p \rightarrow Fq)$

CTL versus LTL

In der LTL lassen sich Fairnesseigenschaften direkt ausdrücken, in der CTL müssen sie im Modellchecking integriert werden.

Umgekehrt gibt es CTL Formeln die Dinge unterscheiden können, die die LTL nicht unterscheiden kann:

Beispiel:
die LTL Formel $\mathbf{F}\boldsymbol{X}\,p$ unterscheidet untenstehende Strukturen nicht,
Die CTL Formel $\mathbf{AF}\boldsymbol{A}\mathbf{X}\,p$ schon!