Temporale Logik

Gebrauch in der Informatik geht auf A. Pnueli zurück

- Häufig zur Spezifikation von Nebenläufigen und Reaktiven System gebraucht
- Erlaubt Verknüpfung von Aussagen zu verschiedenen Zeitpunkten
  - "Morgen ist das Wetter schön"
  - "Die Reaktorstäbe werden nie überhitzt"
  - "Die Zentralverriegelung öffnet sich unmittelbar nach einem Unfall"
  - "Der Airbag löst nur aus, wenn ein Unfall passiert ist"
  - "Einer Bestätigung (Ack) muss eine Anforderung (Req) vorausgehen"
  - "Wenn der Aufzug gerufen wird, dann kommt er auch irgendwann"
- Granularität der Zeitschritte muss natürlich festgelegt werden

Vereinfachte Hennessy-Milner Logik (HML)

zunächst betrachten wir HML als Bsp. für Temporale Logik für LTS

Geg. Alphabet von Aktionen Σ.

Definition Syntax besteht aus booleschen Konstanten $\{0,1\}$, booleschen Operatoren $\{\wedge,\neg,\rightarrow,\ldots\}$ und unären **modalen Operatoren** $[a]$ und $\langle a\rangle$ mit $a \in \Sigma$.

Lese $[a]\,f$ als für alle a-Nachfolger des momentanen Zustandes gilt f

Lese $\langle a\rangle\,f$ als für einen a-Nachfolger des momentanen Zustandes gilt f

Abkürzung $\langle\Theta\rangle\,f$ steht für $\bigvee_{a\in\Theta}\langle a\rangle\,f$ bzw. $[\Theta]\,f$ für $\bigwedge_{a\in\Theta}[a]\,f$

Θ kann auch weiterhin als boolescher Ausdruck über Σ geschrieben werden

z.B. $[a\vee b]\,f \equiv [\{a,b\}]\,f$ oder $\langle\neg a\wedge\neg b\rangle\,f \equiv \langle\Sigma\backslash\{a,b\}\rangle\,f$

Beispiele zur Vereinfachten HML

1.	$[a]\,1$	für alle a-Nachfolger gilt 1 (immer wahr)
2.	$[a]\,0$	für alle a-Nachfolger gilt 0 (a darf nicht möglich sein)
3.	$\langle a\rangle\,1$	für einen a-Nachfolger gilt 1 (es muss a möglich sein)
4.	$\langle a\rangle\,0$	für einen a-Nachfolger gilt 0 (immer falsch)
5.	$\langle a\rangle\,1 \wedge [b]\,0$	es muss a aber es darf nicht b möglich sein
6.	$\langle a\rangle\,1 \wedge [\neg a]\,0$	es muss und darf nur genau a möglich sein
7.	$[a\vee b]\,\langle a\vee b\rangle\,1$	nach a oder b muss wiederum a oder b möglich sein
8.	$\langle a\rangle\,[b]\,[b]\,0$	a ist möglich und danach aber b keinenfalls zweimal
9.	$[a](\langle a\rangle\,1 \rightarrow [a]\,\langle a\rangle\,1)$	ist nach a wiederum möglich, dann auch ein zweitesmal

Semantik der Vereinfachten Hennessy-Milner Logik

Geg. LTS $L = (S, I, \Sigma, T)$.

Definition Semantik ist rekursiv definiert als $s \models f$ (lese "f gilt in s"), mit $s \in S$ und f einer vereinfachte Formel in Hennessy-Milner Logik.

$s \models 1$

$s \not\models 0$

$s \models [\Theta]\,g$ gdw. $\forall\, a \in \Theta\, \forall\, t \in S$: wenn $s \xrightarrow{a} t$ dann $t \models g$

$s \models \langle\Theta\rangle\,g$ gdw. $\exists\, a \in \Theta\, \exists\, t \in S$: $s \xrightarrow{a} t$ und $t \models g$

Definition es gilt $L \models f$ (lese "f gilt in L") gdw. $s \models f$ für alle $s \in I$

Definition Expansion von f ist die Menge der Zustände $[[f]]$ in denen f gilt.

$$[[f]] = \{s \in S \mid s \models f\}$$

Endliche und Unendliche Traces

Geg. LTS $L = (S, I, \Sigma, T)$.

Definitionen

Ein Trace π von L ist eine endliche oder unendliche Folge von Zuständen

$$\pi = (s_0, s_1, \ldots)$$

wobei es für jedes Paar (s_i, s_{i+1}) in π ein $a \in \Sigma$ gibt, mit $s_i \xrightarrow{a} s_{i+1}$. Also gibt es $a_0, a_1, \ldots$ mit

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

$|\pi|$ ist dessen Länge, z.B. $|\pi| = 2$ für $\pi = (s_0, s_1, s_2)$, und $|\pi| = \infty$ für unendliche Traces.

$\pi(i)$ ist der i-te Zustand s_i von π falls $i \leq |\pi|$

$\pi^i = (s_i, s_{i+1}, \ldots)$ ist das Suffix ab und inklusive dem i-ten Zustand s_i falls $i \leq |\pi|$

Bemerkung: $|\pi| = \infty$ dann $|\pi^i| = \infty$ für alle $i \in \mathbb{N}$

Computation Tree Logic für HML (CTL/HML)

zunächst nur in Verbindung mit HML

Definition Syntax aufbauend auf der von HML und zusätzlich

unäre temporale Pfad-Operatoren $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$ und ein binärer temporaler Pfad-Operator $\mathbf{U}$.

Pfad-Operatoren müssen einen Pfad-Quantor $\mathbf{E}$ oder $\mathbf{A}$ als Präfix haben.

$\mathbf{EX}f$	in einem unmittelbaren Folgezustand gilt f	$\equiv \langle\Sigma\rangle f$
$\mathbf{AX}f$	in jedem Nachfolger muss f gelten	$\equiv [\Sigma] f$
$\mathbf{EF}f$	in einer Zukunft gilt f irgendwann	*exists finally*
$\mathbf{AF}f$	in allen möglichen Abläufen gilt f irgendwann	*always finally*
$\mathbf{EG}f$	in einer Zukunft gilt f konstant	*exists globally*
$\mathbf{AG}f$	es gilt f immer	*always globally*
$\mathbf{E}[f\ \mathbf{U}\ g]$	möglicherweise gilt f solange bis schließlich g gilt (uns g muss für diesen Trace irgendwann gelten)	*exists until*
$\mathbf{A}[f\ \mathbf{U}\ g]$	f gilt immer solange bis schließlich g gilt (uns g muss für jeden Trace irgendwann gelten)	*always until*

Beispiele zu CTL/HML

$$\neg\mathbf{EX}f \equiv \mathbf{AX}\neg f \qquad \neg\langle\Theta\rangle f \equiv [\Theta]\neg f \qquad \neg\mathbf{EF}f \equiv \mathbf{AG}\neg f \qquad \neg\mathbf{EG}f \equiv \mathbf{AF}\neg f$$

(De Morgan für $\mathbf{E}[\cdot\ \mathbf{U}\ \cdot]$ benötigt weiteren temporalen Pfad-Operator)

$\mathbf{AG}\,[\neg safe]\,0$	es ist immer unmöglich unsichere Aktionen auszuführen
$\mathbf{EF}\,\langle\neg safe\rangle\,1$	möglicherweise kann unsichere Aktionen ausgeführt werden
$\neg\mathbf{E}[\neg\langle req\rangle\,1\ \mathbf{U}\ \langle ack\rangle\,1]$	es gibt keinen Ablauf auf dem irgendwann ack möglich wird und zuvor req nie möglich war
$\mathbf{AG}\,[req]\,\mathbf{AF}\,[\neg ack]\,0$	immer nach einem req muss ein Punkt erreicht werden, ab dem keine Aktion ausser ack möglich ist

CTL/HML erlaubt die Kombination von Zustands- und Aktionsspezifikation

dies ist aber auch notwendig und leider oftmals unelegant

Semantik der CTL/HML Operatoren

Geg. CTL/HML Formel f, g, ein LTS L. π sei immer ein Trace von L, und $i, j \in \mathbb{N}$.

Definition Semantik $s \models f$ (lese "f gilt in s") ist rekursiv definiert

(nur noch für die neuen CTL Operatoren)

$$s \models \mathbf{EX}f \quad \text{gdw.} \quad \exists\pi[\pi(0) = s \wedge \pi(1) \models f]$$
$$s \models \mathbf{AX}f \quad \text{gdw.} \quad \forall\pi[\pi(0) = s \Rightarrow \pi(1) \models f]$$

$$s \models \mathbf{EF}f \quad \text{gdw.} \quad \exists\pi[\pi(0) = s \wedge \exists i[i \leq |\pi| \wedge \pi(i) \models f]]$$
$$s \models \mathbf{AF}f \quad \text{gdw.} \quad \forall\pi[\pi(0) = s \Rightarrow \exists i[i \leq |\pi| \wedge \pi(i) \models f]]$$

$$s \models \mathbf{EG}f \quad \text{gdw.} \quad \exists\pi[\pi(0) = s \wedge \forall i[i \leq |\pi| \Rightarrow \pi(i) \models f]]$$
$$s \models \mathbf{AG}f \quad \text{gdw.} \quad \forall\pi[\pi(0) = s \Rightarrow \forall i[i \leq |\pi| \Rightarrow \pi(i) \models f]]$$

$$s \models \mathbf{E}[f\ \mathbf{U}\ g] \quad \text{gdw.} \quad \exists\pi[\pi(0) = s \wedge \exists i[i \leq |\pi| \wedge \pi(i) \models g \wedge \forall j[j < i \Rightarrow \pi(j) \models f]]]$$
$$s \models \mathbf{A}[f\ \mathbf{U}\ g] \quad \text{gdw.} \quad \forall\pi[\pi(0) = s \Rightarrow \exists i[i \leq |\pi| \wedge \pi(i) \models g \wedge \forall j[j < i \Rightarrow \pi(j) \models f]]]$$

Kripke Strukturen

- Klassisches Semantisches Modell für Temporale Logik
- reine Zustandssicht, keine Aktionen
  - im Prinzip LTS mit genau einer Aktion $(|\Sigma| = 1)$
  - zusätzlich Annotation von Zuständen mit atomaren Aussagen
- Ursprünge aus der modalen Logik:
  - verschiedene Welten aus S sind über $\rightarrow$ bzw. T verbunden
  - $[\,]f$ gdw. für alle unmittelbar erreichbaren Welten gilt f
  - $\langle\,\rangle f$ gdw. es gibt eine unmittelbar erreichbare Welten, in der f gilt

Kripke Strukturen

Geg. Menge von Atomaren Aussagen $\mathcal{A}$ (boolesche Prädikate).

Definition eine Kripke Struktur $K = (S, I, T, \mathcal{L})$ besteht aus folgenden Komponenten:

- Zustandsmenge S.
- Anfangszuständen $I \subseteq S$ mit $I \neq \emptyset$
- einer *totalen* Übergangsrelation $T \subseteq S \times S$ (T total gdw. $\forall s[\exists t[T(s,t)]]$)
- Labelling/Markierung/Annotation $\mathcal{L}: S \rightarrow \mathbb{P}(\mathcal{A})$.

Labelling bildet Zustand s auf Menge atomarer Aussagen ab, die in s gelten:

$$\mathcal{L}(s) = \{grau, warm, trocken\}$$

LTS als Kripke Struktur

Definition Kripke Struktur $K = (S_K, I_K, T_K, \mathcal{L})$ zu einem vollständigen LTS $L = (S_L, I_L, \Sigma, T_L)$ ist definiert durch folgende Komponenten

$$\mathcal{A} = \Sigma \qquad S_K = S_L \times \Sigma \qquad I_K = I_L \times \Sigma \qquad \mathcal{L}: (s,a) \mapsto a$$

$$T_K((s,a),(s',a')) \quad \text{gdw.} \quad T_L(s,a,s') \text{ und } a' \text{ beliebig}$$

Ähnliche Konstruktion wie beim Orakel-Automat!

Fakt

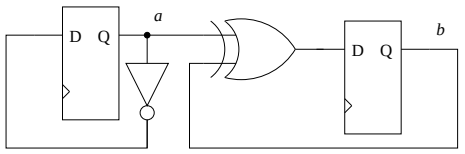
$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n \quad \text{in } L$$

gdw.

$$(s_0,a_0) \rightarrow (s_1,a_1) \cdots \rightarrow (s_n,a_n) \quad \text{in } K$$

Anmerkung oftmals $S \subseteq \mathbb{B}^n$, $\Sigma = \{a_1, \ldots, a_n\}$, und $\mathcal{L}((s_1, \ldots, s_n)) = \{a_i \mid s_i = 1\}$

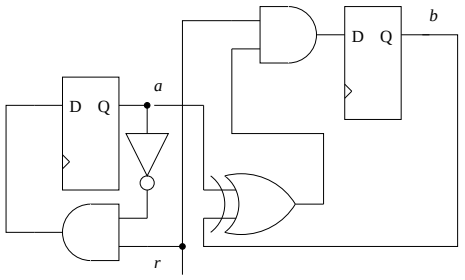
2-Bit Zähler als Kripke-Struktur



$$\begin{aligned} S &= \mathbb{B}^2 \\ I &= \mathbb{B}^2 \\ T &= \{((0,0),(0,1)), \\ &\quad ((0,1),(1,0)), \ldots\} \end{aligned}$$

$a \in L(s)$ gdw. $s \in \{(0,1),(1,1)\}$

$b \in L(s)$ gdw. $s \in \{(1,0),(1,1)\}$



$$\begin{aligned} S &= \mathbb{B}^3 \\ I &= \mathbb{B}^3 \\ T &= \ldots \end{aligned}$$

$a \in L(s)$ gdw. $s \in \{(-,-,1)\}$

$b \in L(s)$ gdw. $s \in \{(-,1,-)\}$

$r \in L(s)$ gdw. $s \in \{(1,-,-)\}$

Netzlisten, also Schaltkreise auf dieser Abstraktionsebene, haben keinen Initialzustand

Computational Tree Logic (CTL)

klassische Version der CTL für Kripke Strukturen

Definition Syntax von CTL enthält alle $p \in \mathcal{A}$, alle booleschen Operatoren $\wedge, \neg, \vee, \rightarrow, \ldots$ und die temporalen Operatoren $\mathbf{EX}, \mathbf{AX}, \mathbf{EF}, \mathbf{AF}, \mathbf{EG}, \mathbf{AG}, \mathbf{E}[\cdot\ \mathbf{U}\ \cdot]$ und $\mathbf{A}[\cdot\ \mathbf{U}\ \cdot]$.

Definition CTL Semantik $s \models f$ (lese "f gilt in s") für Kripke Struktur $K = (S, I, T, \mathcal{L})$ ist genauso rekursiv definiert wie bei CTL/HML wobei zusätzlich $s \models p$ gdw. $p \in \mathcal{L}(s)$.

Beispiele zum 2-Bit Zähler mit Reset

$\mathbf{AG}(\overline{r} \rightarrow \mathbf{AX}(\overline{a} \wedge \overline{b}))$

$\mathbf{AG}\ \mathbf{EX}(\overline{a} \wedge \overline{b})$

$\mathbf{AG}\ \mathbf{EF}(\overline{a} \wedge \overline{b})$

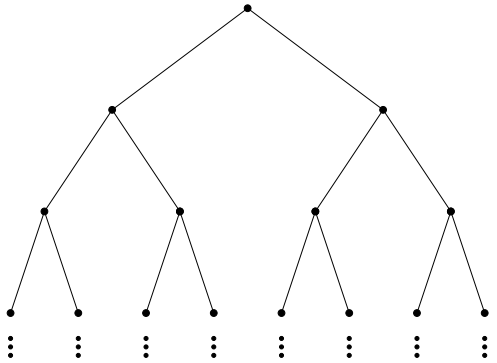
$\mathbf{AG}\ \mathbf{AF}(\overline{a} \wedge \overline{b})$ unendlich oft $\overline{a} \wedge \overline{b}$

$\mathbf{AG}(\overline{a} \wedge \overline{b} \wedge r \rightarrow \mathbf{AX}\ \mathbf{A}[(a \vee b)\ \mathbf{U}\ (\overline{a} \wedge \overline{b})])$

$(\mathbf{AG}\ r) \rightarrow \mathbf{AF}(a \wedge b)$

Definition es gilt f in K schreibe $K \models f$ gdw. $s \models f$ für alle $s \in I$ (generische Definition)

Computation Tree



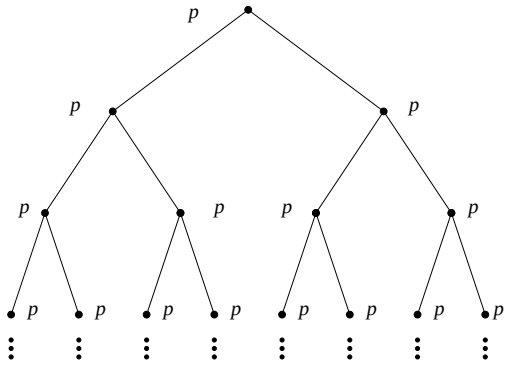
Alle Mögliche Abläufe werden in einem (unendlichen) Baum dargestellt

CTL betrachtet die Verzweigungsstruktur (Branching) des Computation Tree

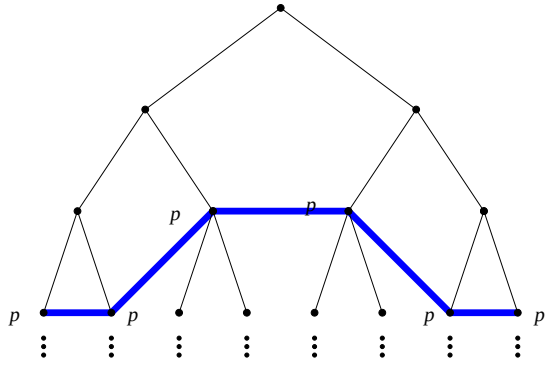
und hat eine lokale Zustandssicht

von jedem betrachteten Zustand verzweigen neue Pfade

Computation Tree $\mathbf{AG}p$



Computation Tree $\mathbf{AF}p$



Linear Temporal Logic (LTL)

Definition Syntax von LTL ist genauso wie die von CTL, nur dass die temporalen Operatoren keine Pfadquantoren besitzen, also aus $\mathbf{X}, \mathbf{F}, \mathbf{G}$ und $\mathbf{U}$ bestehen.

Definition Semantik $\pi \models f$ von LTL Formel f ist auf unendlichen Pfaden π in K definiert:

$\pi \models p$	gdw.	$p \in \mathcal{L}(\pi(0))$
$\pi \models \neg g$	gdw.	$\pi \not\models g$
$\pi \models g \wedge h$	gdw.	$\pi \models g$ und $\pi \models h$
$\pi \models \mathbf{X} g$	gdw.	$\pi^1 \models g$
$\pi \models \mathbf{F} g$	gdw.	$\pi^i \models g$ für ein i
$\pi \models \mathbf{G} g$	gdw.	$\pi^i \models g$ für alle i
$\pi \models g \,\mathbf{U}\, h$	gdw.	es gibt ein i mit $\pi^i \models h$ und $\pi^j \models g$ für alle $j < i$

Definition $K \models f$ gdw. $\pi \models f$ für alle unendlichen Pfade π in K mit $\pi(0) \in I$

Vergleich LTL und CTL

- LTL betrachtet jeweils genau einen möglichen linearen Ablauf.
- damit macht $(\mathbf{G} r) \rightarrow \mathbf{F}(a \wedge b)$ plötzlich Sinn! (Erster Teil Annahme/Einschränkung)
- LTL ist kompositional (bez. synch. Produkt von Kripke-Strukturen):
  - $K_1 \models f_1, K_2 \models f_2 \quad \Rightarrow \quad K_1 \times K_2 \models f_1 \wedge f_2$
  - $K_1 \models f \rightarrow g, K_2 \models f \quad \Rightarrow \quad K_1 \times K_2 \models g$

Fakt CTL und LTL haben unterschiedliche Ausdrucksmächtigkeit:

z.B. lässt sich $\mathbf{AXEX} p$ nicht in LTL ausdrücken, ebenso hat $\mathbf{AFAG} p$ kein LTL Pendant

ACTL Formeln als LTL Formeln

[Clarke and Draghicescu'88]

ACTL ist Teilmenge von CTL ohne $\mathbf{E}$ Pfadquantor und Negation nur vor $p \in \mathcal{A}$.

Definition zu einer ACTL Formel f definiere $f \backslash \mathbf{A}$ als die LTL Formel, die aus f durch Wegstreichen aller $\mathbf{A}$ Pfadquantoren entsteht.

Definition f und g sind äquivalent gdw. $K \models f \Leftrightarrow K \models g$ für alle Kripke-Strukturen K.

(f und g können aus unterschiedlichen Logiken stammen)

Satz falls ACTL Formel f zu LTL Formel g äquivalent, dann auch zu $f \backslash \mathbf{A}$.

Beweis $K \models f \overset{\text{Annahme}}{\Leftrightarrow} \forall \pi[\pi \models g] \overset{\text{Annahme}}{\underset{+\text{s.u.}}{\Leftrightarrow}} \forall \pi[\pi \models f] \overset{!}{\Leftrightarrow} \forall \pi[\pi \models f \backslash \mathbf{A}] \overset{\text{Def.}}{\Leftrightarrow} K \models f \backslash \mathbf{A}$

(π immer initialisiert und in $\pi \models f$ als Kripkestruktur interpretiert)

Syntaktisch Charakterisierte Teilmenge von LTL und CTL

[M. Maidl'00]

Seien f und g bel. CTL bzw. LTL Formeln und $p \in \mathcal{A}$.

Definition Jede Unterformel einer CTL$^{\text{det}}$ Formel hat eine der folgenden Formen:

$$p, \quad f \wedge g, \quad \mathbf{AX} f, \quad \mathbf{AG} f, \quad (\neg p \wedge f) \vee (p \wedge g) \quad \text{oder} \quad \mathbf{A}[(\neg p \wedge f) \,\mathbf{U}\, (p \wedge g)]$$

Definition Jede Unterformel einer LTL$^{\text{det}}$ Formel hat eine der folgenden Formen:

$$p, \quad f \wedge g, \quad \mathbf{X} f, \quad \mathbf{G} f, \quad (\neg p \wedge f) \vee (p \wedge g) \quad \text{oder} \quad (\neg p \wedge f) \,\mathbf{U}\, (p \wedge g)$$

Satz Schnittmenge von LTL und ACTL besteht aus LTL$^{\text{det}}$ bzw. CTL$^{\text{det}}$

Intuition CTL-Semantik bei CTL$^{\text{det}}$ beschränkt sich auf Auswahl genau eines Pfades

Hinweis $\mathbf{A}[f \,\mathbf{U}\, p] \equiv \mathbf{A}[(\neg p \wedge f) \,\mathbf{U}\, (p \wedge 1)] \qquad \mathbf{AF} p \equiv \mathbf{A}[1 \,\mathbf{U}\, p]$

$\Rightarrow$ eine nicht-deterministische Spezifikation birgt Gefahren der Falsch-Interpretation

LTL und CTL Können nicht Zählen

[P. Wolper'83]

Spezifikation "jeden m-ten Schritt gilt p" (zumindest)

Fakt für alle $m > 1$ gibt es weder eine CTL noch LTL Formel f, mit

$K \models f$ gdw. $\pi(i) \models p$ für alle initialisierten Pfade π von K und alle $i = 0 \bmod m$.

Problem $p \wedge \mathbf{G}(p \leftrightarrow \neg \mathbf{X} p)$ bedeutet "genau jeden 2. Schritt gilt p"

Lösungen

- modulo m Zähler ins Modell integrieren (Schwierigkeiten mit Kompositionalität)
- Erweiterung der Logik
  - ETL mit zusätzlichen Temporalen Operatoren definiert durch Automaten ...
  - ... bzw. Quantoren über atomaren Variablen (damit Zähler in der Logik)
  - reguläre Ausdrücke: $\neg \left((\underbrace{1;\ldots;1}_{m-1};p)^*;\underbrace{1;\ldots;1}_{m-1};\neg p \right)$ bzw. $(\underbrace{1;\ldots;1}_{m-1};p)^\omega$

Fairness

- Spezifikation mach oft nur Sinn unter Fairness-Annahmen
  - z.B. Abstraktion des Schedulers: "jeder Prozess kommt dran"
  - z.B. eine Komponente muss unendlich oft am Zuge sein
  - z.B. der Übertragungskanal produziert unendlich oft keinen Fehler
- kein Problem in LTL: $(\mathbf{GF} f) \rightarrow \mathbf{G}(r \rightarrow \mathbf{F} a)$
- Faire Kripke-Strukturen für CTL:
  - zusätzliche Komponente F von fairen Zuständen
  - ein Pfad π ist fair gdw. $|\{i \mid \pi(i) \in F\}| = \infty$
  - betrachte nur noch faire Pfade

μ-Kalkül

- spezielle Form von Quantoren über Mengen von Zuständen
  - quantifizierte Variablen $V = \{X, Y, \ldots\}$
  - i.Allg. auch für Mengen und damit Logik zweiter Ordnung
- Fixpunkt-Logik: kleinste Fixpunkte spezifiziert durch μ und größte durch ν
- Modaler μ-Kalkül als Erweiterung von HML bzw. CTL

$$\nu X[p \wedge [\,]X] \equiv \mathbf{AG} p \qquad \mu X[q \vee (p \wedge \langle\,\rangle X)] \equiv \mathbf{E}[p\,\mathbf{U}\,q]$$

$\nu X[p \wedge [\,][\,]X]$ entspricht "jeden 2. Schritt gilt p"

$\nu X[p \wedge \langle\,\rangle \mu Y[(f \wedge X) \vee (p \wedge \langle\,\rangle Y)]] \equiv \nu X[p \wedge \mathbf{EXE}[p\,\mathbf{U}\,f \wedge X]] \equiv \mathbf{EG} p$ unter Fairness f

Semantik μ-Kalkül

Auch wieder über Kripke Struktur $K = (S, I, T, \mathcal{L})$.

Definition eine Belegung ρ über den Variablen V ist eine Abb. $\rho : V \rightarrow \mathbb{P}(S)$

Definition Semantik $[[f]]_\rho$ einer μ-Kalkül Formel f ist rekursiv definiert als Expansion, also als Menge Zustände in denen f für eine geg. Belegung ρ gilt:

$$\begin{array}{rclrcl} [[p]]_\rho & = & \{s \mid p \in \mathcal{L}(s)\} & [[X]]_\rho & = & \rho(X) \\ [[\neg f]]_\rho & = & S \backslash [[f]]_\rho & [[f \wedge g]]_\rho & = & [[f]]_\rho \cap [[g]]_\rho \\ \mu X[f] & = & \bigcap \{A \subseteq S \mid [[f]]_{\rho[X \mapsto A]} = A\} & \nu X[f] & = & \bigcup \{A \subseteq S \mid [[f]]_{\rho[X \mapsto A]} = A\} \end{array}$$

mit $\rho[A \mapsto X](Y) = \begin{cases} A & X = Y \\ \rho(Y) & X \neq Y \end{cases}$.

Definition $K \models f$ gdw. $I \subseteq [[f]]_\rho$ für alle Belegungen ρ

Fakt μ-Kalkül subsumiert LTL und CTL.