Kapitel II

Temporallogik und Model Checking

Inhalt Kapitel II

- Einführung
- Die Temporallogik CTL
  - Syntax und informelle Semantik
  - Semantik
  - Äquivalenzen
- CTL-Model Checking
  - Labelling Algorithmus
  - Optimierungen
- Das System SMV
- Fairness
- Das Alternating Bit Protokoll
- Symbolisches Model-Checking
- Bounded Model-Checking

Motivation

Unter *Model Checking* versteht man die automatische Überprüfung, ob ein Systemmodell eine Spezifikation erfüllt.

Die Modellierungen nebenläufiger Systeme aus Kapitel 1 waren bereits Beispiele dafür:

- Die Modellierungen mit SAT-Solvern sind Instanzen von *Bounded Model Checking* (da die Simulationszeit beschränkt ist).
- Die Modellierungen mit BDDs sind Instanzen von *Symbolic Model Checking* (da Zustandsmengen nicht explizit, sondern “symbolisch” repräsentiert wurden)

Arten von Eigenschaften

Typische Arten von spezifizierten Eigenschaften:

- **Safety**: System gerät in keinen “verbotenen” Zustand / alle erreichbaren Zustände sind “erlaubt” (hatten wir schon).
- **Liveness**: System verklemmt sich nicht; “Reset-Zustand” von überall ereichbar; jede “Anfrage” wird irgendwann “beantwortet”.
- **Fairness**: bestimmte “gute Eigenschaft” gilt für alle “fairen” Abläufe.

Diese Klassifikation erfasst die meisten Eigenschaften, bisweilen gibt es noch komplexere.

Temporallogik

Temporallogik erlaubt die kompakte Spezifikation von Eigenschaften von Systemabläufen.

Im Unterschied zur Aussagenlogik können auch Aussagen über den zeitlichen Ablauf gemacht werden.

Es gibt eine Reihe verschiedener Temporallogiken, z.B.

- CTL (Computation Tree Logic)
- LTL (Linear Time Logic)

In der Vorlesung wird CTL im Detail behandelt.

Temporallogik

Im Semaphorbeispiel haben wir überprüft, dass das System keinen unerwünschten Zustand erreichen kann.

In CTL kann das durch folgende Formel ausgedrückt werden.

$$\mathrm{AG}(\neg \mathit{undesired})$$

Diese Formel sagt aus, dass alle (A – all) Abläufe im Zustandsübergangssystem stets (G – generally) die Eigenschaft $\neg \mathit{undesired}$ erfüllen.

Temporallogik

Die Eigenschaft, dass stets wieder der Anfangszustand erreicht werden kann, kann in CTL wie folgt ausgedrückt werden:

$$\mathsf{AG}(\mathsf{EF}(\bigwedge_p q_{p\ sleep}))$$

Die Formel $\mathsf{EF}(\phi)$ besagt, dass ein Ablauf existiert (E – exists), auf dem irgendwann (F – finally) die Eigenschaft ϕ gilt.

Syntax von CTL

Die Menge der CTL-Formeln ist durch folgende Grammatik gegeben.

$$\begin{aligned}\phi, \psi ::= \; & p \mid \top \mid \bot \mid \neg\phi \mid \phi \oplus \psi \mid \mathsf{AX}\phi \mid \mathsf{EX}\phi \\ & \mid \mathsf{A}[\phi \mathsf{U} \psi] \mid \mathsf{E}[\phi \mathsf{U} \psi] \mid \mathsf{AG}\phi \mid \mathsf{AF}\phi \mid \mathsf{EG}\phi \mid \mathsf{EF}\phi\end{aligned}$$

Hier steht p für aussagenlogische Variablen und $\oplus$ steht für die zweistelligen Boole'schen Operatoren. Insbesondere ist also jede aussagenlogische Formel auch eine CTL-Formel.

Beispiel: $\mathsf{AG}(p \Rightarrow \mathsf{A}[p\mathsf{U}(\neg p \wedge \mathsf{A}[\neg p \mathsf{U} q])])$

Kein Beispiel: $\mathsf{A}[p]$ und $\phi\mathsf{U}\psi$ sind keine CTL-Formeln!

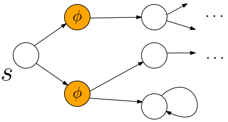
Informelle Semantik der CTL-Formeln

CTL Formeln werden relativ zu einem gegebenen Zustandsübergangssystem interpretiert.

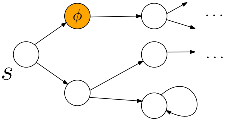
Eine CTL-Formel ϕ kann in jedem Zustand entweder gelten (= wahr sein) oder nicht.

In einem Zustand s gilt...

- ...AXϕ, wenn ϕ in allen unmittelbaren Folgezuständen von s gilt.

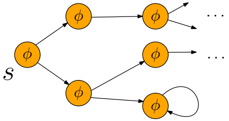


- ...EXϕ, wenn ϕ in einem der unmittelbaren Folgezustände von s gilt.

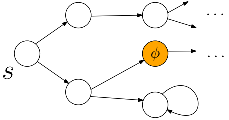


Informelle Semantik der CTL-Formeln, Forts.

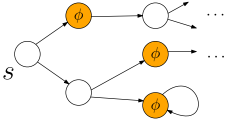
- …AGϕ, wenn ϕ auf allen von s aus erreichbaren Zuständen gilt.



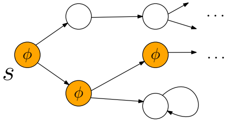
- …EFϕ, wenn man von s aus einen Zustand erreichen kann, in dem ϕ gilt.



- …AF(ϕ), wenn auf allen von s ausgehenden Ausführungspfaden irgendwann ϕ gilt.

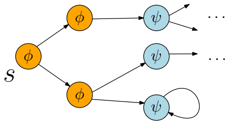


- …EG(ϕ), wenn von s aus die Ausführung so fortgesetzt werden kann, dass stets ϕ gilt.

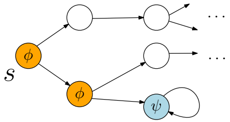


Informelle Semantik der CTL-Formeln, Forts.

- …A[ϕUψ], wenn auf allen von s ausgehenden Ausführungspfaden irgendwann ψ gilt und zumindest bis zum ersten Auftreten von ψ stets ϕ der Fall ist. (U = "until").



- …E[ϕUψ], wenn von s aus die Ausführung so fortgesetzt werden kann, dass irgendwann ψ gilt und bis dahin stets ϕ gilt.



Informelle Semantik der CTL-Formeln, Forts.

Beispiele:

- $\mathrm{AG}((\mathit{close\_door} \vee (\mathit{safe} \wedge \neg \mathit{open\_door})) \Rightarrow \mathrm{AX}\mathit{safe}) \wedge \mathrm{AG}(\mathit{heat} \Rightarrow \mathit{safe})$
- $\mathit{floor}{=}2 \wedge \mathit{direction}{=}\mathit{up} \wedge \mathit{buttonpressed}{=}5 \Rightarrow A[\mathit{direction}{=}\mathit{up}\ \mathrm{U}\ \mathit{floor}{=}5]$
- $\mathrm{AF}\mathit{fertig}$

Transitionssystem

Definition

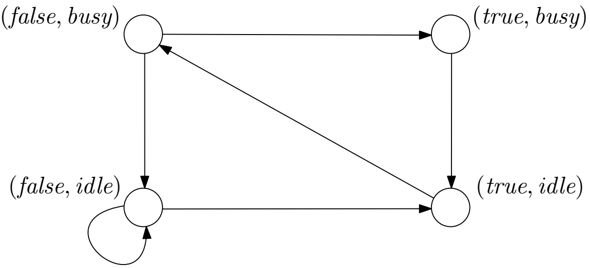
Ein *Transitionssystem* ist ein Paar $(S, \rightarrow)$, wobei

- S eine Menge von Zuständen ist, und
- $\rightarrow \subseteq S \times S$ eine binäre Relation auf S ist.
- Für jedes $s \in S$ existiert $s' \in S$ mit $s \rightarrow s'$.

- Die Menge S modelliert die Menge der globalen Zustände eines nebenläufigen Systems.
- Die Relation $\rightarrow$ heißt *Transitionsrelation*. Sie modelliert die möglichen Zustandsübergänge. Sie ergibt sich aus dem Programmtext, bzw. der Implementierung des Systems.
- Die dritte Bedingung hat technische Gründe. Liegt sie nicht bereits vor, so kann sie durch Hinzunahme eines Müllzustands s_d mit $s_d \rightarrow s_d$ künstlich hergestellt werden.

Beispiel

$$\begin{aligned}
S &= \{(\mathit{request}, \mathit{status}) \mid \mathit{request} \in \{\mathit{true}, \mathit{false}\}, \mathit{status} \in \{\mathit{idle}, \mathit{busy}\}\} \\
\rightarrow &= \{((\mathit{false}, x), (\mathit{true}, x)) \mid x \in \{\mathit{idle}, \mathit{busy}\}\} \cup \\
&\quad \{((\mathit{true}, \mathit{idle}), (\mathit{false}, \mathit{busy}))\} \cup \\
&\quad \{((x, \mathit{busy}), (x, \mathit{idle})) \mid x \in \{\mathit{true}, \mathit{false}\}\} \cup \\
&\quad \{((\mathit{false}, \mathit{idle}), (\mathit{false}, \mathit{idle}))\}
\end{aligned}$$



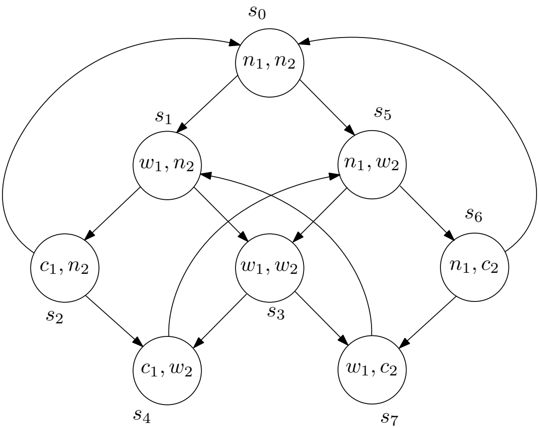
Weitere Beispiele

- Semaphor: $S = \{(\mathit{proc}_0, \mathit{proc}_1, \mathit{sem}) \mid \mathit{sem} \in \{\mathit{free}, \mathit{occ}\}, \forall i \in \{0,1\}.\mathit{proc}_i \in \{\mathit{sleep}, \mathit{wait}, \mathit{work}\}\}$
  Hier: $|S| = 18$
- Peterson:
  $S = \{(\mathit{flag}_0, \mathit{flag}_1, \mathit{turn}, \mathit{line}_0, \mathit{line}_1) \mid \forall i{\in}\{0,1\}.\mathit{flag}_i \in \{\mathit{true}, \mathit{false}\} \;\&\; \mathit{turn} \in \{0,1\} \;\&\; \mathit{line}_i \in \{0,1,2,3,4\}\}$
  Hier: $|S| = 2^2 \cdot 2 \cdot 5^2 = 200$
- Bauer, Hund, Katze, Maus: $S = \{(\mathit{pos}_B, \mathit{pos}_H, \mathit{pos}_K, \mathit{pos}_M) \mid \forall x \in \{B,H,K,M\}.\ \mathit{pos}_x \in \{\mathit{links}, \mathit{rechts}\}\}$
  Hier: $|S| = 2^4 = 16$.

NB: Die Transitionsrelation $\rightarrow$ ist hier jeweils weggelassen.

Weitere Beispiele

Der von Zustand $s_0 = (\mathit{sleep}, \mathit{sleep}, \mathit{free})$ aus erreichbare Teil des Semaphor-Transitionssystems:



Formale Semantik von CTL

Die Semantik von CTL-Formeln wird bezüglich einer Interpretation festgelegt.

Definition

Eine *Interpretation* $\mathcal{I}$ besteht aus einem endlichen Transitionssystem *Tr*$(\mathcal{I}) = (S, \rightarrow)$ sowie einer Menge von Zuständen $\mathcal{I}(p) \subseteq S$ für jede aussagenlogische Variable p.

Sei $\mathcal{I}$ eine Interpretation $\mathcal{I}$ mit *Tr*$(\mathcal{I}) = (S, \rightarrow)$.

Die CTL-Semantik legt für jede Formel ϕ und jeden Zustand $s \in S$ fest, ob die Formel in diesem Zustand bezüglich der Interpretation $\mathcal{I}$ gilt.

Wir schreiben kurz $s \models_{\mathcal{I}} \phi$ für "ϕ gilt im Zustand s (bezüglich $\mathcal{I}$)" und definieren diesen Begriff auf den nächsten Folien.

Definition der Semantik

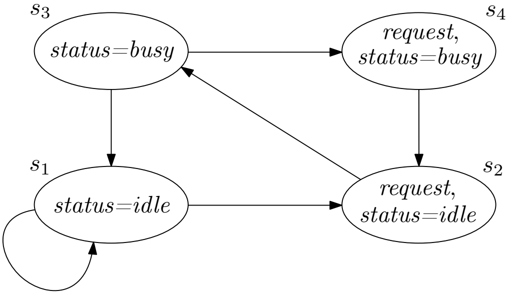
- $s \models_{\mathcal{I}} p$ genau dann wenn $s \in \mathcal{I}(p)$.
- $s \models_{\mathcal{I}} \neg\phi$ genau dann wenn $s \models_{\mathcal{I}} \phi$ nicht gilt (auch geschrieben als $s \not\models_{\mathcal{I}} \phi$).
- $s \models_{\mathcal{I}} \phi \wedge \psi$ genau dann wenn $s \models_{\mathcal{I}} \phi$ und $s \models_{\mathcal{I}} \psi$.
- die anderen Boole'schen Operatoren $\vee$, $\Rightarrow$, etc. sind analog.
- $s \models_{\mathcal{I}} \mathsf{EX}\phi$ genau dann wenn $s' \in S$ existiert mit $s \to s'$ und $s' \models_{\mathcal{I}} \phi$.
- $s \models_{\mathcal{I}} \mathsf{AX}\phi$ genau dann wenn für alle $s' \in S$ mit $s \to s'$ gilt: $s' \models_{\mathcal{I}} \phi$.

Definition der Semantik, Fortsetzung

- $s \models_{\mathcal{I}}$ AGϕ gdw: Alle unendlichen Pfade der Form $s = s_0 \to s_1 \to s_2 \to s_3 \to \dots$ haben die Eigenschaft, dass $s_i \models_{\mathcal{I}} \phi$ für alle $i \geqslant 0$ gilt.
- $s \models_{\mathcal{I}}$ EGϕ gdw: Es gibt einen unendlichen Pfad der Form $s = s_0 \to s_1 \to s_2 \to s_3 \to \dots$ mit der Eigenschaft, dass $s_i \models_{\mathcal{I}} \phi$ für alle $i \geqslant 0$ gilt.
- $s \models_{\mathcal{I}}$ EFϕ gdw: Es gibt einen unendlichen Pfad der Form $s = s_0 \to s_1 \to s_2 \to s_3 \to \dots$ mit der Eigenschaft, dass $s_i \models_{\mathcal{I}} \phi$ für ein $i \geqslant 0$ gilt.
- $s \models_{\mathcal{I}}$ AFϕ gdw: Alle unendlichen Pfade der Form $s = s_0 \to s_1 \to s_2 \to s_3 \to \dots$ haben die Eigenschaft, dass $s_i \models_{\mathcal{I}} \phi$ für ein $i \geqslant 0$ gilt.

Beispiel

Die Interpretation $\mathcal{I}$ mit dem Transitionssystem von Folie 120 sowie $\mathcal{I}(\textit{request}) = \{s_2, s_4\}$, $\mathcal{I}(\textit{status=idle}) = \{s_1, s_2\}$ und $\mathcal{I}(\textit{status=busy}) = \{s_3, s_4\}$ wird folgendermaßen dargestellt.



Es gilt:

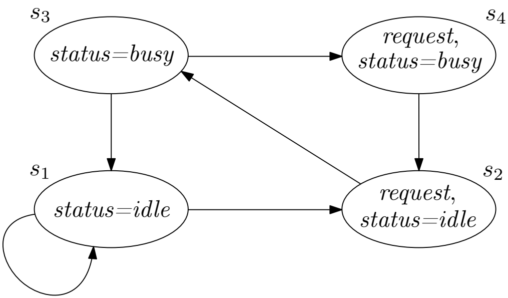
$s_1 \models_{\mathcal{I}} \mathsf{AF}\neg\textit{request}$ $\qquad$ $s_1 \models_{\mathcal{I}} \mathsf{AG}(\textit{request} \Rightarrow \mathsf{EF}(\textit{status=busy}))$

$s_1 \models_{\mathcal{I}} \mathsf{EG}\neg\textit{request}$ $\qquad$ $s_1 \models_{\mathcal{I}} \mathsf{AG}(\neg\mathsf{EG}(\textit{status=busy}))$

Semantik der Until-Formeln

- $s \models_{\mathcal{I}} \mathsf{E}[\phi \mathsf{U} \psi]$ gdw: Es gibt einen Pfad $s = s_0 \to s_1 \to s_2 \to s_3 \to \cdots \to s_n$ mit der Eigenschaft, dass dass $s_n \models_{\mathcal{I}} \psi$ gilt sowie dass $s_i \models_{\mathcal{I}} \phi$ für alle $i < n$ gilt.
- $s \models_{\mathcal{I}} \mathsf{A}[\phi \mathsf{U} \psi]$ gdw: Alle unendlichen Pfade $s = s_0 \to s_1 \to s_2 \to s_3 \to \ldots$ haben die Eigenschaft, dass ein $n \geqslant 0$ existiert mit $s_n \models_{\mathcal{I}} \psi$ und $s_i \models_{\mathcal{I}} \phi$ für alle $i < n$.

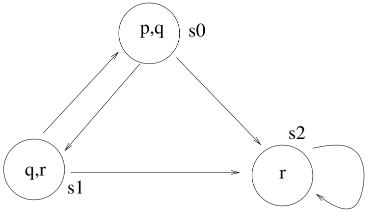
Beispiel



Es gilt:

$$s_1 \models_{\mathcal{I}} \mathsf{AG}(\mathit{request} \Rightarrow \mathsf{A}[\mathit{request}\,\mathsf{U}\,\mathit{status{=}busy}])$$

Beispiel



$$
\begin{array}{ll}
s_0 \models_{\mathcal{I}} p \wedge q & s_0 \models_{\mathcal{I}} p \wedge \neg r \\
s_0 \models_{\mathcal{I}} \top & s_0 \models_{\mathcal{I}} \mathsf{EX}(q \wedge r) \\
s_0 \models_{\mathcal{I}} \neg\mathsf{AX}(q \wedge r) & s_0 \models_{\mathcal{I}} \neg\mathsf{EF}(p \wedge r) \\
s_1 \models_{\mathcal{I}} \mathsf{EG}r & s_2 \models_{\mathcal{I}} \mathsf{AG}r \\
s_0 \models_{\mathcal{I}} \mathsf{AF}r & s_0 \models_{\mathcal{I}} \mathsf{E}[(p \wedge q)\mathsf{U}r] \\
s_0 \models_{\mathcal{I}} \mathsf{A}[p\mathsf{U}r] &
\end{array}
$$

Äquivalenzen

Äquivalenz von CTL-Formeln

Zwei CTL-Formeln ϕ und ψ sind *äquivalent*, geschrieben $\phi \iff \psi$, wenn für alle Interpretationen $\mathcal{I}$ und alle Zustände s gilt: $s \models_{\mathcal{I}} \phi$ gdw. $s \models_{\mathcal{I}} \psi$.

Sind $\phi \iff \psi$ aussagenlogisch äquivalente Formeln, so auch als CTL-Formeln. Z.B.: $\mathsf{AG}(p) \vee \mathsf{AG}(p) \iff \mathsf{AG}(p)$.

Wichtige Äquivalenzen:

$$\neg\mathsf{AG}(\phi) \iff \mathsf{EF}(\neg\phi) \qquad \neg\mathsf{AF}(\phi) \iff \mathsf{EG}(\neg\phi)$$

$$\neg\mathsf{EF}(\phi) \iff \mathsf{AG}(\neg\phi) \qquad \neg\mathsf{EG}(\phi) \iff \mathsf{AF}(\neg\phi)$$

$$\mathsf{AF}(\phi) \iff \mathsf{A}[\top \mathsf{U} \phi] \qquad \mathsf{EF}(\phi) \iff \mathsf{E}[\top \mathsf{U} \phi]$$

$$\mathsf{A}[\phi \mathsf{U} \psi] \iff \neg(\mathsf{E}[\neg\psi \mathsf{U} (\neg\phi \wedge \neg\psi)] \vee \mathsf{EG}(\neg\psi))$$

Äquivalenzen

Satz

Für jede CTL-Formel ϕ gibt es eine äquivalente Formel, in der neben Variablen nur die Operatoren $\neg$, $\wedge$, $\bot$, EX, AF, E[$-$U$-$] verwendet werden.

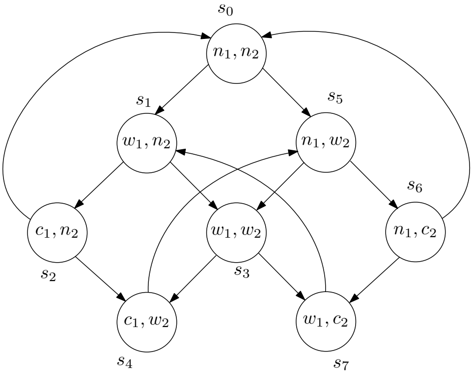
Beweis: Übung

Das Model Checking Problem für CTL

Gegeben:
Eine Interpretation $\mathcal{I}$ und eine CTL-Formel ϕ_0 und ein Zustand s_0.

Gefragt:
Gilt $s_0 \models_{\mathcal{I}} \phi_0$?

Mutual Exclusion Beispiel



- Safety: AG$(\neg(c_1 \wedge c_2))$
- Liveness: AG$(w_1 \Rightarrow$ AF$c_1)$
- Non-blocking: AG$(n_1 \Rightarrow$ EX$w_1)$
- No strict sequencing: EF$(c_1 \wedge$ E$[c_1$U$(\neg c_1 \wedge$ E$[\neg c_2$U$c_1])])$

Labelling Algorithmus

Der *Labelling Algorithmus* löst das Model Checking für CTL.

- Eingabe sind eine Interpretation $\mathcal{I}$ und eine Formel ϕ_0.
- Berechnet wird die Menge aller Zustände (im Transitionssystem von $\mathcal{I}$) in denen ϕ_0 gilt.

Aufgrund der Äquivalenzen können wir annehmen, dass die Formel ϕ_0 nur die Operatoren $\neg$, $\wedge$, $\bot$, EX, AF, E$[-$U$-]$ verwendet.

Labelling Algorithmus

Grundidee: Berechne für jede Teilformel von ϕ von ϕ_0 die Menge aller Zustände, in denen ϕ gilt.

- Bildlich gesprochen beschriftet (labelt) man die Zustände in S mit denjenigen Teilformeln, die dort gelten.
- Der Algorithmus verfährt durch Rekursion über die Formel.
  Für Variablen ist die Aufgabe einfach.
  Bei einer zusammengesetzten Formel ϕ führt man den Algorithmus zunächst für die direkten Teilformeln aus. Aus dem Ergebnis kann man dann die Beschriftung für ϕ berechnen.

Labelling Algorithmus: Details (1)

Der Algorithmus macht eine Fallunterscheidung über die Eingabeformel ϕ_0.

- $\perp$: Markiere keinen Zustand mit $\perp$.
- p: Markiere Zustände mit aussagenlogischen Variablen, wie von der Interpretation vorgegeben.
- $\neg\phi$: Führe den Algorithmus rekursiv für ϕ aus. Markiere danach alle Zustände mit $\neg\phi$, die nicht mit ϕ beschriftet sind.
- $\phi \wedge \psi$: Führe den Algorithmus rekursiv für ϕ und ψ aus. Markiere danach alle Zustände mit $\phi \wedge \psi$, die sowohl mit ϕ als auch mit ψ beschriftet sind.
- EXϕ: Führe den Algorithmus rekursiv für ϕ aus. Markiere dann alle Zustände mit EXϕ, die einen unmittelbaren Nachfolger haben, der schon mit ϕ markiert ist.

Labelling Algorithmus: Details (2)

- AFϕ:
  1. Führe den Algorithmus rekursiv für ϕ aus.
  2. Markiere alle Zustände mit AFϕ, die schon mit ϕ markiert sind.
  3. Sind *alle* unmittelbaren Folgezustände eines Zustands s bereits mit AFϕ markiert, so markiere auch s mit AFϕ. Wiederhole Schritt 3 bis keine neuen Markierungen mehr hinzukommen.
- E$[\phi$U$\psi]$:
  1. Führe den Algorithmus rekursiv für ϕ und ψ aus.
  2. Markiere alle Zustände mit E$[\phi$U$\psi]$, die schon mit ψ markiert sind.
  3. Ist *ein* unmittelbarer Folgezustände eines Zustands s bereits mit E$[\phi$U$\psi]$ markiert und ist s selbst mit ϕ markiert, so markiere s auch mit E$[\phi$U$\psi]$. Wiederhole Schritt 3 bis keine neuen Markierungen mehr hinzukommen.

Komplexität

Eine direkte Implementierung des Algorithmus hat Laufzeit

$$O(f \cdot V \cdot (V + E))$$

wobei f die Größe der Ausgangsformel, V die Zahl der Zustände und E die Zahl der Transitionen ist.

Beispiel: AFϕ

1. rekursiver Aufruf: $O((f-1) \cdot V \cdot (V + E))$
2. Anfangsmarkierung: $O(V)$
3. einen Zustand, dessen Nachfolger alle schon markiert sind, finden und markieren: $O(V + E)$; maximal V Wiederholungen

Verbesserung

Labelling kann in Zeit $O(f \cdot (V + E))$ implementiert werden.

Dazu genügt es, alle Fälle so zu implementieren, dass zum rekursiven Aufruf jeweils nur Aufwand $O(V + E)$ hinzukommt.

- Die Fälle für p, $\neg$, $\wedge$, EX sind einfach.
- Der Fall für E$[\phi$U$\psi]$ kann mit einer Rückwärts-Breitensuche implementiert werden.
  Ist ein Knoten mit E$[\phi$U$\psi]$ markiert, so werden alle seine Vorgänger, die auch mit ϕ markiert sind, selbst mit $E[\phi$U$\psi]$ markiert.
- Leider funktioniert Breitensuche für AF nicht, da ja alle Nachfolger und nicht nur einer markiert sein müssen.

Effiziente Behandlung von EGϕ

Anstatt ein effizienteres Verfahren für AF direkt anzugeben, ersetzen wir AF durch EG und geben ein Verfahren für EG an.

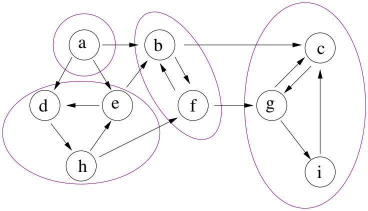
Beachte: AF$\phi \iff \neg$EG$(\neg\phi)$

Starke Zusammenhangskomponenten

Definition

Eine *starke Zusammenhangskomponente* (strongly connected component, SCC) eines gerichteten Graphs ist eine maximal große Menge U von Knoten mit folgender Eigenschaft: Für alle $s_1, s_2 \in U$ gilt, dass s_2 von s_1 aus erreichbar ist und umgekehrt.

Eine SCC ist *trivial*, wenn sie aus einem einzigen Knoten besteht, der keine Kante zu sich selbst hat.



Ein Graph kann in linearer Zeit in SCCs zerlegt werden (Tarjan's Alg.).

Effiziente Behandlung von EGϕ

Markierung der Zustände mit EGϕ.

- Führe den Algorithmus rekursiv für ϕ aus.
- Betrachte folgenden Teilgraphen G des Transitionsystems: Die Knoten sind alle mit ϕ markierten Zustände. Zwischen diesen Knoten hat G die gleichen Kanten wie das Transitonssystem.
- Berechne die SCCs von G.
- Entferne aus G alle trivialen SCCs.
- Markiere in G alle Knoten, von denen aus eine verbleibende SCC erreichbar ist.
- Markiere im ursprünglichen Transitionssystem alle Zustände, die in G markiert sind.

Korrektheit der Markierung mit EGϕ

Angenommen der rekursive Aufruf für ϕ markiert genau die Zustände mit ϕ, in denen ϕ erfüllt ist.

Alle mit EGϕ markierten Zustände erfüllen die Formel EGϕ.

- Für jeden markierten Zustand gibt es in G einen Pfad in eine nichttrivale SCC von G.
- Ist s ein Knoten in einer nichttrivialen SCC, dann gibt es einen unendlichen Pfad $s \to s_0 \to s_1 \to \ldots$, der in der SCC bleibt.
- Es gibt also von jedem mit EGϕ markierten Knoten aus einen unendlichen Pfad in G.
- Da G nur mit ϕ markierte Knoten enthält, folgt daraus, das EGϕ erfüllt ist.

Vollständigkeit der Markierung mit EGϕ (1)

Alle Zustände, die EGϕ erfüllen, werden auch markiert.

Dazu ist zu zeigen, dass jeder Zustand s markiert wird, der $s \models_{\mathcal{I}}$ EGϕ erfüllt. Diese Eigenschaft gilt genau dann wenn es einen unendlichen Pfad $s = s_0 \to s_1 \to \ldots$ gibt, dessen Zustände alle mit ϕ markiert sind.

Lemma

In jedem unendlichen Pfad $s_0 \to s_1 \to s_2 \to \ldots$ in einem endlichen Graphen kommen ab einem bestimmten Index n nur noch Zustände vor, die auch unendlich oft im Pfad vorkommen.

Beweis: Für jeden Zustand s, der im Pfad nur endlich oft vorkommt, gibt es einen Index n_s des letzten Vorkommens. Sei n das Maximum aller dieser n_s (es gibt nur endlich viele, da S endlich ist). Ab Index n können nach Konstruktion nur Zustände vorkommen, die auch unendlich oft im Pfad vorkommen. $\square$

Vollständigkeit der Markierung mit EGϕ (2)

Satz

Für jeden unendlichen Pfad $s_0 \to s_1 \to s_2 \to \ldots$ in einem endlichen Graphen gibt es eine nichttriviale SCC U und eine Zahl n, so dass $s_i \in U$ für alle $i \geqslant n$ gilt.

Beweis: Wähle n nach dem Lemma. Für beliebige $n \leqslant i < j$ können wir nun zeigen, dass s_i und s_j in derselben SCC liegen. Wegen $i < j$ gibt es einen Pfad von s_i nach s_j. Es gibt auch einen Pfad von s_j nach s_i. Es muss es ein $k > j$ geben mit $s_k = s_i$, denn andernfalls käme s_i nur endlich oft im Pfad vor. Somit ist s_i von s_j erreichbar und umgekehrt; sie liegen in einer SCC. Die SCC enthält eine Kante und ist daher nichttrivial. □

Nach dem Satz wird also jeder Zustand s markiert, für den es einen unendlichen Pfad $s = s_0 \to s_1 \to \ldots$ gibt, dessen Zustände alle mit ϕ markiert sind. Das sind alle Zustände, die EGϕ erfüllen.

State-Explosion-Problem

Die effizientere Version ist linear in der Größe des Transitionssystems, aber…

Die Größe des Transitionssystems ist exponentiell in der Anzahl seiner Komponenten: n Prozesse à k Zustände ergeben ein Transitionssystem mit k^n Zuständen.

Abhilfen:

- Ausnutzen von Symmetrie
- Abstraktion
- Symbolische Repräsentation von Zuständen (wie bei der Modellierung mit BDDs im ersten Kapitel)