

## 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 erreichbar; 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.

$$AG(\neg \textit{undesired})$$

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

# Temporallogik

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

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

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

# Syntax von CTL

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

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

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

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



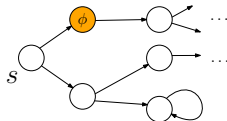
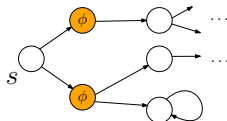
# Informelle Semantik der CTL-Formeln

CTL Formeln werden relativ zu einem gegebenen Zustandsübergangssystem interpretiert.

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

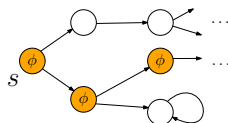
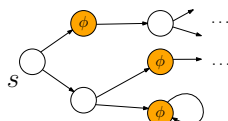
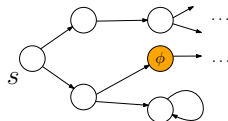
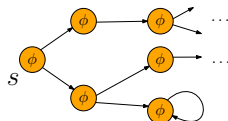
In einem Zustand  $s$  gilt...

- ... $AX\phi$ , wenn  $\phi$  in allen unmittelbaren Folgezuständen von  $s$  gilt.
- ... $EX\phi$ , wenn  $\phi$  in einem der unmittelbaren Folgezustände von  $s$  gilt.



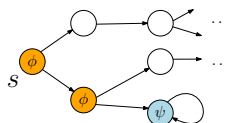
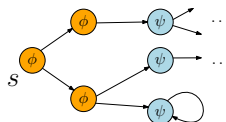
# Informelle Semantik der CTL-Formeln, Forts.

- ... $AG\phi$ , wenn  $\phi$  auf allen von  $s$  aus erreichbaren Zuständen gilt.
- ... $EF\phi$ , wenn man von  $s$  aus einen Zustand erreichen kann, in dem  $\phi$  gilt.
- ... $AF(\phi)$ , wenn auf allen von  $s$  ausgehenden Ausführungspfaden irgendwann  $\phi$  gilt.
- ... $EG(\phi)$ , wenn von  $s$  aus die Ausführung so fortgesetzt werden kann, dass stets  $\phi$  gilt.



# Informelle Semantik der CTL-Formeln, Forts.

- ... $A[\phi U \psi]$ , wenn auf allen von  $s$  ausgehenden Ausführungspfaden irgendwann  $\psi$  gilt und zumindest bis zum ersten Auftreten von  $\psi$  stets  $\phi$  der Fall ist. (U = “until”).
- ... $E[\phi U \psi]$ , wenn von  $s$  aus die Ausführung so fortgesetzt werden kann, dass irgendwann  $\psi$  gilt und bis dahin stets  $\phi$  gilt.



# Informelle Semantik der CTL-Formeln, Forts.

## Beispiele:

- $AG((close\_door \vee (safe \wedge \neg open\_door)) \Rightarrow AXsafe) \wedge AG(heat \Rightarrow safe)$
- $floor=2 \wedge direction=up \wedge buttonpressed=5 \Rightarrow A[direction=up \cup floor=5]$
- $AFfertig$

# Transitionssystem

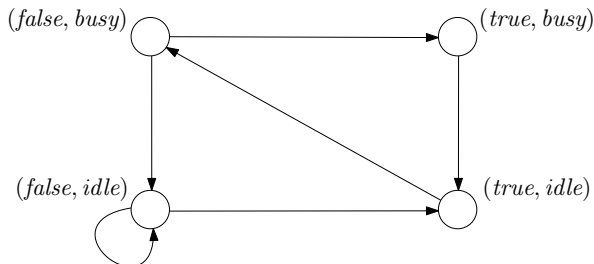
## Definition

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

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



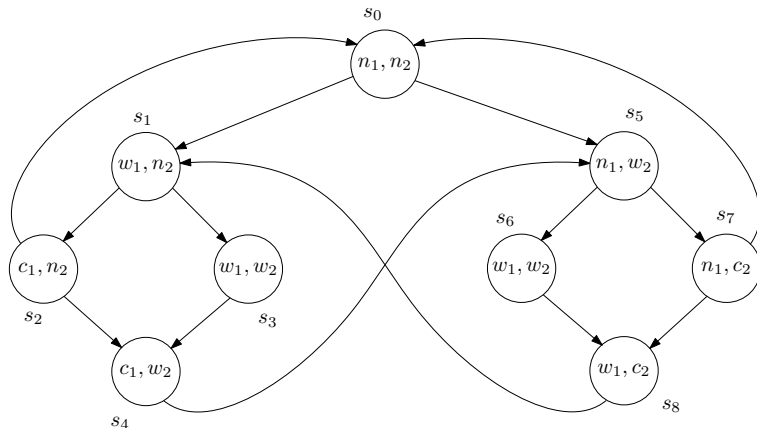
## Weitere Beispiele

- Semaphore:  $S = \{(proc_0, proc_1, sem) \mid sem \in \{free, occ\}, \forall i \in \{0, 1\}. proc_i \in \{sleep, wait, work\}\}$   
Hier:  $|S| = 18$
- Peterson:  
 $S = \{(flag_0, flag_1, turn, line_0, line_1) \mid \forall i \in \{0, 1\}. flag_i \in \{true, false\} \ \& \ turn \in \{0, 1\} \ \& \ line_i \in \{0, 1, 2, 3, 4\}\}$   
Hier:  $|S| = 2^2 \cdot 2 \cdot 5^2 = 200$
- Bauer, Hund, Katze, Maus:  $S = \{(pos_B, pos_H, pos_K, pos_M) \mid \forall x \in \{B, H, K, M\}. pos_x \in \{links, rechts\}\}$   
Hier:  $|S| = 2^4 = 16$ .

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

# Weitere Beispiele

Der von Zustand  $s_0 = (\text{sleep}, \text{sleep}, \text{free})$  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  $\phi$  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 “ $\phi$  gilt im Zustand  $s$  (bezüglich  $\mathcal{I}$ )” und definieren diesen Begriff auf den nächsten Folien.

# Definition der Semantik

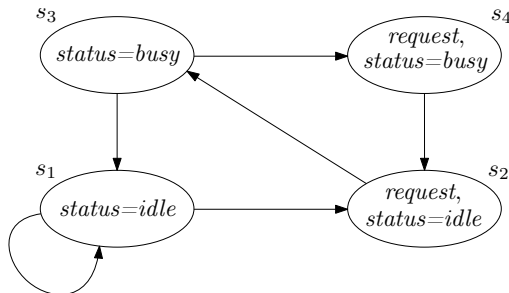
- $s \models_{\mathcal{I}} p$  genau dann wenn  $s \in \mathcal{I}(p)$ .
- $s \models_{\mathcal{I}} \neg\phi$  genau dann wenn  $s \models_{\mathcal{I}} \phi$  nicht gilt (auch geschrieben als  $s \not\models_{\mathcal{I}} \phi$ ).
- $s \models_{\mathcal{I}} \phi \wedge \psi$  genau dann wenn  $s \models_{\mathcal{I}}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

## Definition der Semantik, Fortsetzung

- $s \models_{\mathcal{I}} \text{AG}\phi$  gdw: Alle unendlichen Pfade  $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$  haben die Eigenschaft, dass  $s_i \models_{\mathcal{I}} \phi$  für alle  $i \geq 0$  gilt.
- $s \models_{\mathcal{I}} \text{EG}\phi$  gdw: Es gibt einen unendlichen Pfad  $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$  mit der Eigenschaft, dass  $s_i \models_{\mathcal{I}} \phi$  für alle  $i \geq 0$  gilt.
- $s \models_{\mathcal{I}} \text{EF}\phi$  gdw: Es gibt einen unendlichen Pfad  $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$  mit der Eigenschaft, dass  $s_i \models_{\mathcal{I}} \phi$  für ein  $i \geq 0$  gilt.
- $s \models_{\mathcal{I}} \text{AF}\phi$  gdw: Alle unendlichen Pfade  $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$  haben die Eigenschaft, dass  $s_i \models_{\mathcal{I}} \phi$  für ein  $i \geq 0$  gilt.

# Beispiel

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



Es gilt:

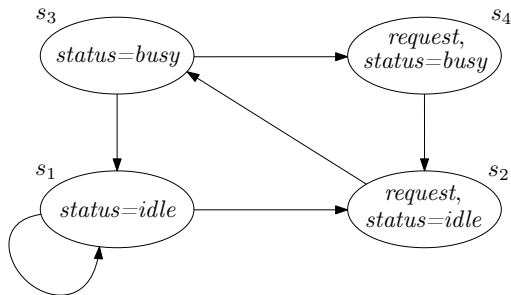
$$s_1 \models_{\mathcal{I}} \text{AF} \neg \text{request} \quad s_1 \models_{\mathcal{I}} \text{AG}(\text{request} \Rightarrow \text{EF}(\text{status=busy}))$$

$$s_1 \models_{\mathcal{I}} \text{EG} \neg \text{request} \quad s_1 \models_{\mathcal{I}} \text{AG}(\neg \text{EG}(\text{status=busy}))$$

## Semantik der Until-Formeln

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

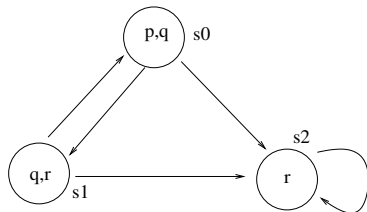
# Beispiel



Es gilt:

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

# Beispiel



$$s_0 \models_{\mathcal{I}} p \wedge q$$

$$s_0 \models_{\mathcal{I}} \top$$

$$s_0 \models_{\mathcal{I}} \neg \text{AX}(q \wedge r)$$

$$s_1 \models_{\mathcal{I}} \text{EG}r$$

$$s_0 \models_{\mathcal{I}} \text{AF}r$$

$$s_0 \models_{\mathcal{I}} \text{A}[p\text{Ur}]$$

$$s_0 \models_{\mathcal{I}} p \wedge \neg r$$

$$s_0 \models_{\mathcal{I}} \text{EX}(q \wedge r)$$

$$s_0 \models_{\mathcal{I}} \neg \text{EF}(p \wedge r)$$

$$s_2 \models_{\mathcal{I}} \text{AG}r$$

$$s_0 \models_{\mathcal{I}} \text{E}[(p \wedge q)\text{Ur}]$$

# Äquivalenzen

## Äquivalenz von CTL-Formeln

Zwei CTL-Formeln  $\phi$  und  $\psi$  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.:  $AG(p) \vee AG(p) \iff AG(p)$ .

### Wichtige Äquivalenzen:

$$\neg AG(\phi) \iff EF(\neg\phi)$$

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

$$\neg EF(\phi) \iff AG(\neg\phi)$$

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

$$AF(\phi) \iff A[\top U \phi]$$

$$EF(\phi) \iff E[\top U \phi]$$

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



# Äquivalenzen

## Satz

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

Beweis: Übung

# Das Model Checking Problem für CTL

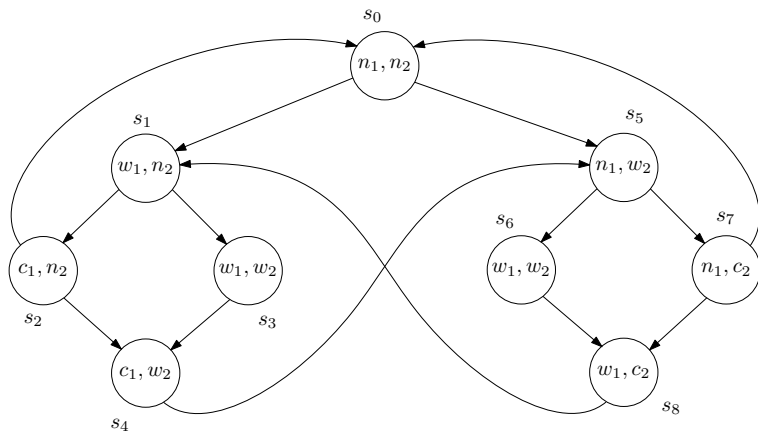
**Gegeben:**

Eine Interpretation  $\mathcal{I}$  und eine CTL-Formel  $\phi_0$  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:  $AG(w_1 \Rightarrow AFc_1)$
- Non-blocking:  $AG(n_1 \Rightarrow EXw_1)$
- 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  $\phi_0$ .
- Berechnet wird die Menge aller Zustände (im Transitionssystem von  $\mathcal{I}$ ) in denen  $\phi_0$  gilt.

Aufgrund der Äquivalenzen können wir annehmen, dass die Formel  $\phi_0$  nur die Operatoren  $\neg$ ,  $\wedge$ ,  $\perp$ , EX, AF, E[ $-U-$ ] verwendet.

# Labelling Algorithmus

**Grundidee:** Berechne für jede Teilformel von  $\phi$  von  $\phi_0$  die Menge aller Zustände, in denen  $\phi$  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  $\phi$  führt man den Algorithmus zunächst für die direkten Teilformeln aus. Aus dem Ergebnis kann man dann die Beschriftung für  $\phi$  berechnen.

# Labelling Algorithmus: Details (1)

Der Algorithmus macht eine Fallunterscheidung über die Eingabeformel  $\phi_0$ .

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

## Labelling Algorithmus: Details (2)

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

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  $\phi$  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 $EG\phi$

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

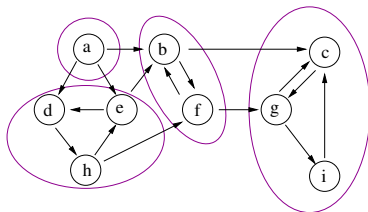
Beachte:  $AF\phi \iff \neg EG(\neg\phi)$

# 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.



# Effiziente Behandlung von $EG\phi$

Markierung der Zustände mit  $EG\phi$ .

- Führe den Algorithmus rekursiv für  $\phi$  aus.
- Betrachte folgenden Teilgraphen  $G$  des Transitionssystems:  
Die Knoten sind alle mit  $\phi$  markierten Zustände. Zwischen diesen Knoten hat  $G$  die gleichen Kanten wie das Transitionssystem.
- 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 $EG\phi$

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

Alle mit  $EG\phi$  markierten Zustände erfüllen die Formel  $EG\phi$ .

- Für jeden markierten Zustand gibt es in  $G$  einen Pfad in eine nichttriviale SCC von  $G$ .
- Ist  $s$  ein Knoten in einer nichttrivialen SCC, dann gibt es einen unendlichen Pfad  $s \rightarrow s_0 \rightarrow s_1 \rightarrow \dots$ , der in der SCC bleibt.
- Es gibt also von jedem mit  $EG\phi$  markierten Knoten aus einen unendlichen Pfad in  $G$ .
- Da  $G$  nur mit  $\phi$  markierte Knoten enthält, folgt daraus, dass  $EG\phi$  erfüllt ist.

# Vollständigkeit der Markierung mit $EG\phi$ (1)

Alle Zustände, die  $EG\phi$  erfüllen, werden auch markiert.

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

## Lemma

In jedem unendlichen Pfad  $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$  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. □

## Vollständigkeit der Markierung mit $EG\phi$ (2)

### Satz

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

Beweis: Wähle  $n$  nach dem Lemma. Für beliebige  $n \leq i < j$  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  $s = s_0 \rightarrow s_1 \rightarrow \dots$  gibt, dessen Zustände alle mit  $\phi$  markiert sind. Das sind alle Zustände, die  $EG\phi$  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  $\rightarrow k$  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)