LTL und CTL*-Model Checking

H. Peter Gumm

Philipps-Universität Marburg

Sommersemester 2007

Lineare Temporale Logik

- LTL ist
  - einfacher zu verstehen als CTL
  - Kann Fairness-Eigenschaften ausdrücken
- LTL-Model Checking ist aufwändiger
  - etwas schwieriger zu verstehen
  - Komplexität: Exponentiell in der Größe der Formel
  - In der Praxis ist das unproblematisch
- Oft mit Hilfe von CTL-Modelchecker überprüfbar

Positive Normalform

- In aussagenlogischen Formeln kann man immer Negationen nach innen bringen
  - $\neg(\neg x \wedge (\neg(y \vee z) \rightarrow u)) = x \vee ((y \vee z) \wedge \neg u)$
  - Negationen nach innen bringen
  - Doppelte Negation entfernen
- Verantwortliche Gleichungen:
  - $\neg(\varphi \vee \psi) = \neg\varphi \wedge \neg\psi$
  - $\neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi$
  - $\neg(\varphi \rightarrow \psi) = \varphi \wedge \neg\psi$
  - $\neg\neg\varphi = \varphi$
- Geht das auch bei temporalen Formeln ?
  - $\neg X \varphi = X \neg \varphi$
  - $\neg F \varphi = G \neg \varphi$
  - $\neg G \varphi = F \neg \varphi$
  - aber ...
- $\neg (\varphi U \psi) = ???$

Releases

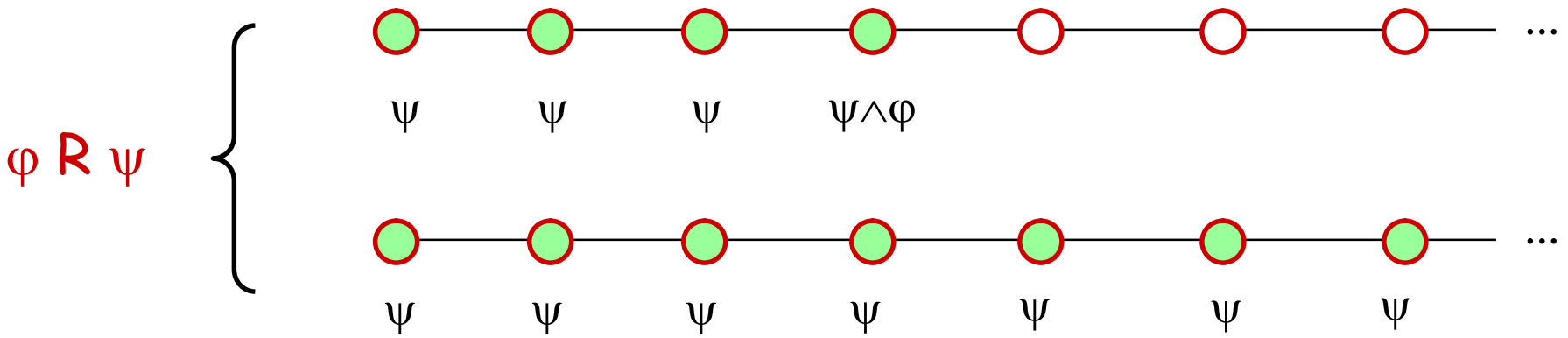
- Führe einen neuen temporalen Operator ein
  - $\varphi \; R \; \psi \;:= \neg(\neg \varphi \; U \; \neg \psi)$
- Es gilt dann
  - $\neg(\varphi \; U \; \psi) = \neg\varphi \; R \; \neg\psi$
  - $\neg(\varphi \; R \; \psi) = \neg\varphi \; U \; \neg\psi$
- Erweitere LTL um R
- Jede Formel lässt sich in PNF (positive Normalform) bringen:
  - Negationen innen,
  - Teilformeln ohne temporale Operatoren in DNF
- R heißt „releases" -- *entlässt* / löst ab
  - aber was *bedeutet* $\varphi \; R \; \psi$ ... ?

Fixpunkt-Gleichungen

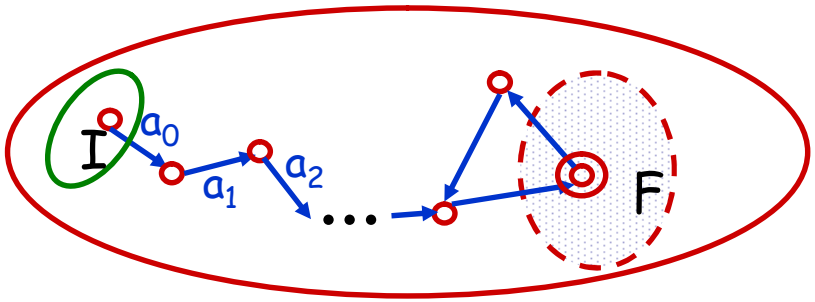
- $F\varphi = \varphi \vee X\, F\varphi$
- $G\varphi = \varphi \wedge X\, G\varphi$
- $\varphi\ U\ \psi = \psi \vee (\varphi \wedge X(\varphi\ U\ \psi))$
- $\varphi\ R\ \psi = \psi \wedge (\varphi \vee X(\varphi\ R\ \psi))$

Herleitung:

$$\begin{aligned}
\varphi\ R\ \psi &= \neg(\neg\,\varphi\ U\ \neg\psi\ ) \\
&= \neg(\neg\psi \vee (\neg\,\varphi \wedge X(\neg\,\varphi\ U\ \neg\psi\ ))) \\
&= (\psi \wedge (\varphi \vee \neg\, X(\,\neg\,\varphi\ U\ \neg\psi\ ))) \\
&= (\psi \wedge (\varphi \vee X\ \neg(\,\neg\,\varphi\ U\ \neg\psi\ ))) \\
&= (\psi \wedge (\varphi \vee X(\,\varphi\ R\ \psi\ )))
\end{aligned}$$



Büchi-Automat



Endlicher Automat für unendliche Worte

$B = (S, \Sigma, \delta, I, F)$

S	endliche Menge von Zuständen
$I \subseteq S$	Anfangszustände
Σ	endliches Alphabet
$\delta \subseteq S \times \Sigma \times S$	nichtdeterministische Transitionsrelation
F	eine Menge von *akzeptierenden Zuständen*

Lauf für ein unendliches S-Wort $w = a_0 a_1 a_2 \ldots \in \Sigma^\omega$:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} s_4 \xrightarrow{a_4} \ldots$$

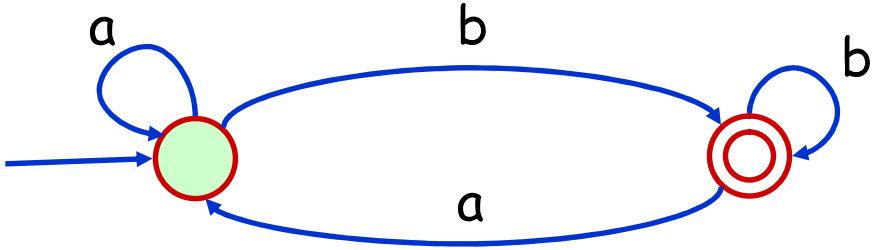
Wort $w=a_0a_1a_2\ldots$ ist in der Sprache L(B), falls es einen *akzeptierenden Lauf* $s_0, s_1, s_2, \ldots$ gibt mit

- $s_0 \in I$
- $(s_i, a_i, s_{i+1}) \in \delta$
- unendlich viele der s_i aus F.

Beispiele

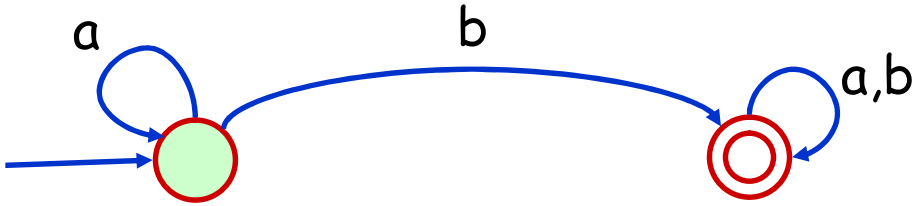
Beispiele von Büchi-Automaten mit zwei Zuständen und $\Sigma = \{a,b\}$.

A_1:

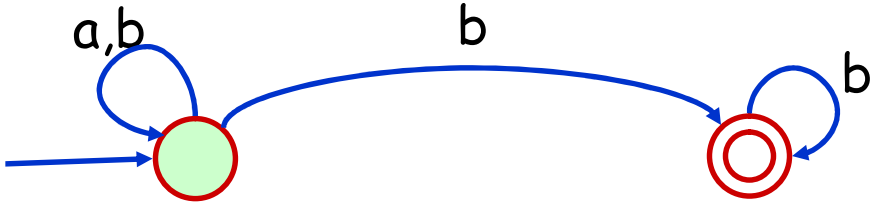


$L(A_1) = GF\ b$

A_2:



$L(A_2) = a\ U\ b = F\ b$

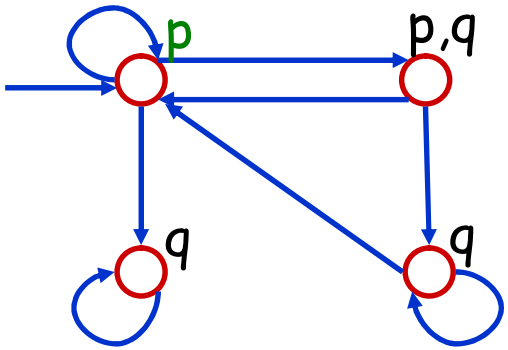


A_3:

$L(A_3) = FG\ b$

Kripke-Struktur als Büchi-Automat

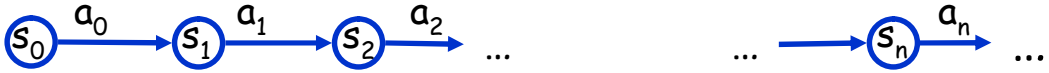
- K=(S,R,L) Kripke-Struktur über AP.
  - $I \subseteq S$ eine beliebige Menge von Anfangszuständen.
  - Setze $\Sigma := 2^{AP}$.
  - Spur einer Berechnung ist unendliches Wort $\tau \in \Sigma^{\omega}$.
- Sprache der Kripke Struktur :
  - $L(K) := \{ \tau \mid \tau \text{ Spur einer Berechnung} \} \subseteq \Sigma^{\omega}$
- Behauptung: Es gibt einen Büchi-Automaten B_K mit $L(B_K) = L(K)$.
  - $B=(S,I,\delta,2^{AP},S)$
  - $(s,a,s') \in \delta \quad :\Leftrightarrow \quad L(s) = a \text{ und } s\ R\ s'$
  - $F = S$ -- jeder Zustand ist akzeptierend.

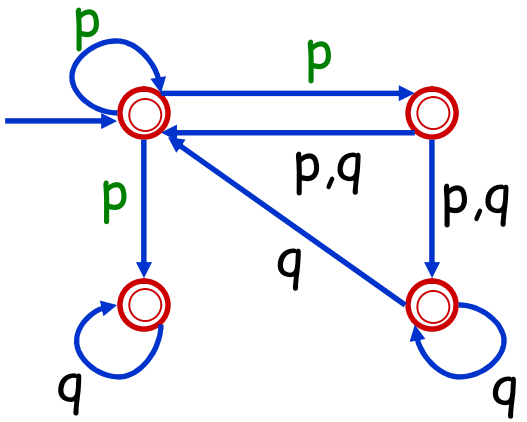


Kripke-Struktur

Kripke

Büchi





Büchi-Automat

Produkt von Büchi-Automaten(1)

- Produkt von Büchi-Automaten soll Schnitt der Sprachen liefern

$B_1=(S_1,\Sigma,\delta_1,I_1,F_1)$ und $B_2=(S_2,\Sigma,\delta_2,I_2,F_2)$ Büchi-Automaten

Erste Idee:

$$B_1 \times B_2 := (S_1 \times S_2,\ \Sigma,\ \delta_1 \times \delta_2,\ I_1 \times I_2)$$

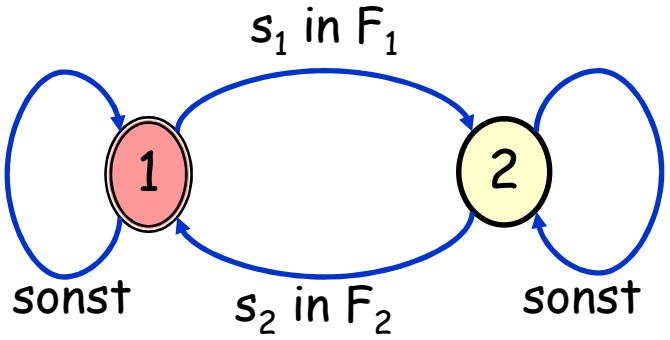
und δ komponentenweise, d.h.:

$$((s_1,s_2),a,(s_1',s_2')) \in \delta_1 \times \delta_2 \quad :\Leftrightarrow \quad (s_1,a,s_1') \in \delta_1 \text{ und } (s_2,a,s_2') \in \delta_2.$$

Akzeptierende Zustände ?
Man will erreichen:

- unendlich oft linke Komponente in F_1
- unendlich oft rechte Komponente in F_2

Idee: Parallelschaltung mit diesem Automaten :



Produkt von Büchi-Automaten(2)

- Produkt von Büchi-Automaten soll Schnitt der Sprachen liefern

$B_1=(S_1,\Sigma,\delta_1,I_1,F_1)$ und $B_2=(S_2,\Sigma,\delta_2,I_2,F_2)$ Büchi-Automaten

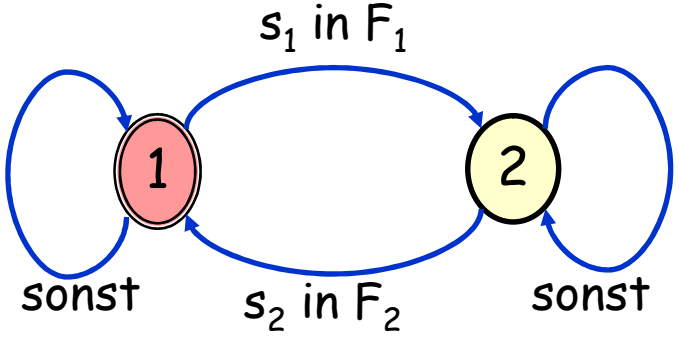
$B_1 \times B_2 := (S_1 \times S_2 \times \{1,2\}, \Sigma, \delta, I_1 \times I_2 \times \{0,1\})$

und δ komponentenweise, d.h.:

$((s_1,s_2,\mathbf{i}),a,(s_1',s_2',\mathbf{j})) \in \delta \quad :\Leftrightarrow$

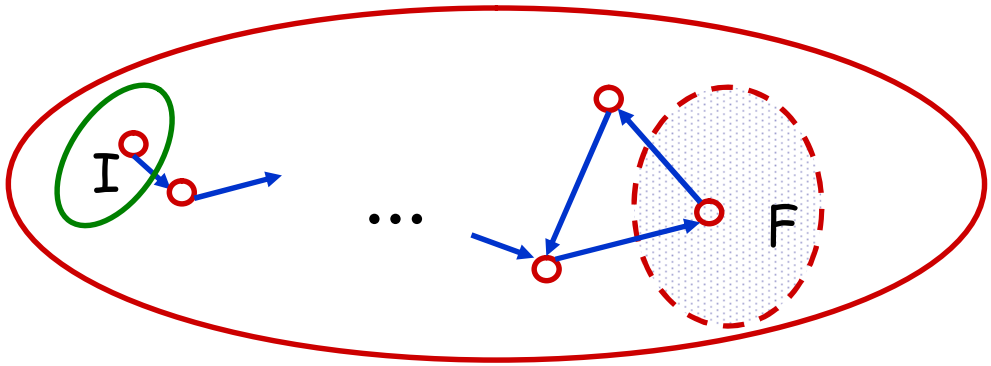
$(s_1,a,s_1') \in \delta_1$ und
$(s_2,a,s_2') \in \delta_2$ und
$\mathbf{j} = 2$, falls $s_1 \in F_1$
$\mathbf{j} = 1$, falls $s_2 \in F_2$
$i = j$, sonst.

Akzeptierende Zustände: $F := F_1 \times S_2 \times \{1\}$



L(B) = ∅ ?

- Es gibt einen effizienten Algorithmus, um festzustellen, ob L(B) = ∅ ist:
  - Bestimme alle SCCs (strongly connected components) C
  - Entferne alle, für die C∩F=∅.
  - Prüfe, ob es einen Pfad von einem Anfangszustand in eine der Cs gibt



Benötigt: Eine Schleife, die einen akzeptierenden Zustand enthält und einen Weg von einem Anfangszustand zu dieser Schleife.

Korollar: Es gibt einen Algorithmus, der entscheidet, ob $L(B_1) \cap L(B_2) = \emptyset$.

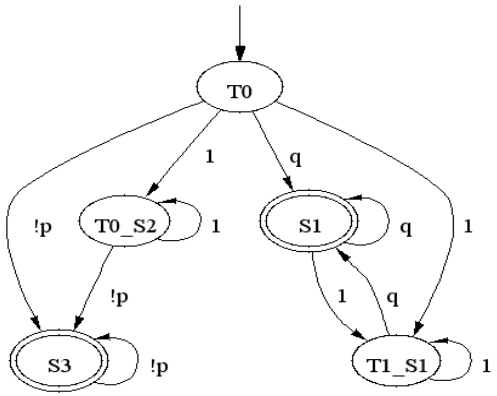
Büchi-Automaten für LTL

- Satz(Wolper): Zu jeder LTL-Formel φ über AP gibt es einen Büchi-Automaten $B\varphi$ mit Alphabet $\Sigma=2^{AP}$, so daß für jede Spur τ gilt:

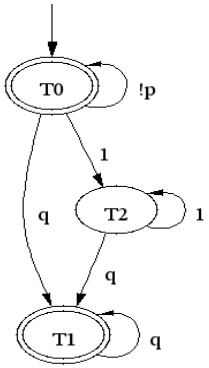
$$\tau \models \varphi \Leftrightarrow \tau \in L(B\varphi)$$

Beispiele:

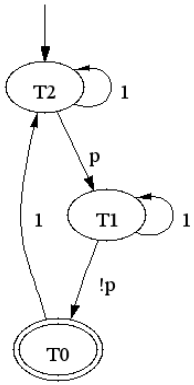
$G\,F\,p \rightarrow G\,F\,q$



$G\,(\,p \rightarrow FG\,q)$

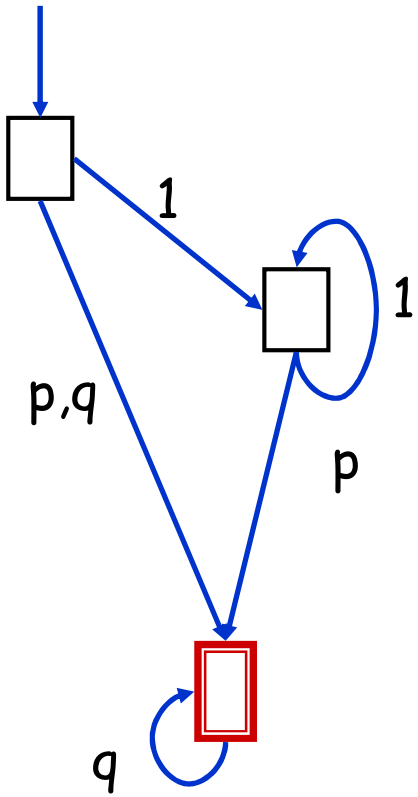


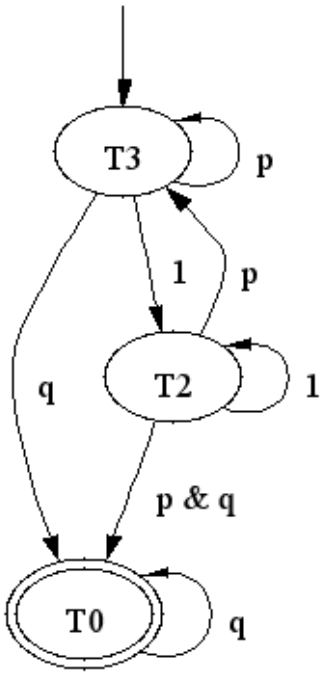
$G\,F\,p \wedge G\,F\,\neg p$

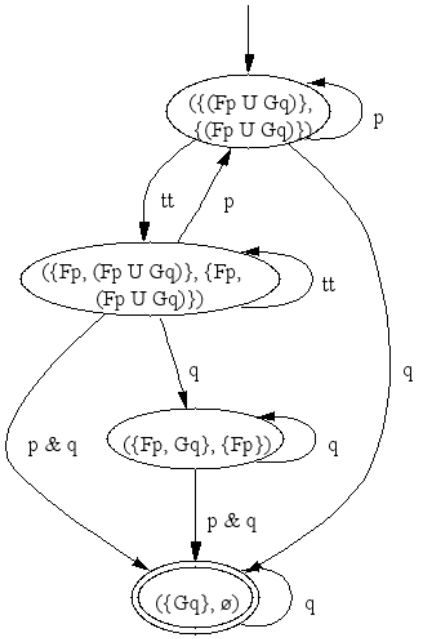


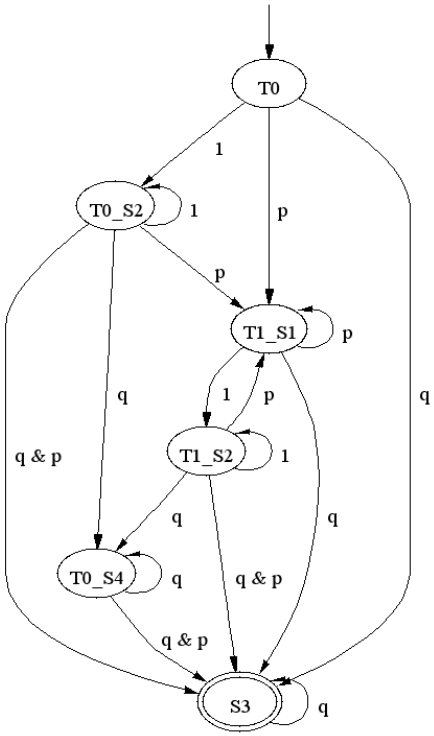
Beispiele erzeugt mit http://www.ti.informatik.uni-kiel.de/~teegen/ABA-Simulation/ltl.cgi

Minimaler Büchi-Automat für (Fp U Gq)





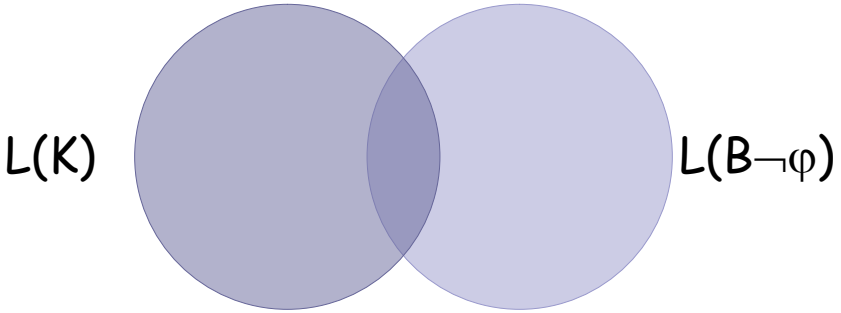




Von http://www.ti.informatik.uni-kiel.de/~teegen/ABA-Simulation/ltl.cgi erzeugte Automaten

LTL-Model Checking-Algorithmus

- Kripke-Struktur K=(S,R,L),
  - evtl. Anfangszustände $I \subseteq S$.
- LTL Formel φ
  - Jede Berechnung von K erfüllt φ
  - $\Leftrightarrow L(K) \subseteq L(B\varphi)$
  - $\Leftrightarrow L(K) \cap L(B\neg\varphi) = \varnothing$.



- Folglich:
  - Konstruiere $B\neg\varphi$ Büchi-Automat für $\neg\varphi$. Akzeptierende Menge $F\neg\varphi$.
  - Fasse K als Büchi-Automat auf und bilde $K \times B\neg\varphi$
  - Bestimme alle SCCs, die die akzeptierenden Zustände schneiden
  - Gibt es einen Pfad τ von einem Anfangszustand zu einer der Cs ?
    - Nein : K erfüllt φ.
    - Ja : K erfüllt φ nicht

    und
    - der gefundene Pfad liefert ein Gegenbeispiel

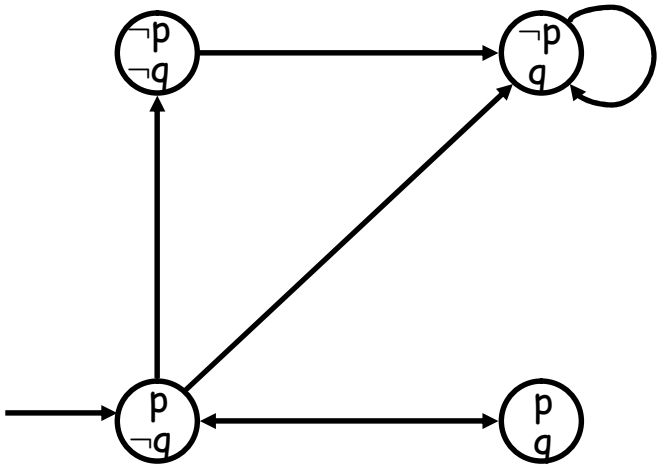
LTL-Formeln mit CTL-Model Checker

- NuSMV ist ein CTL Model Checker
  - Wieso kann man damit LTL-Formeln prüfen ?

- Gegeben
  - Kripke-Struktur K
  - LTL-Formel φ
- Konstruiere Tableau $T_{\neg\varphi}$ für $\neg\varphi$
  - Allgemeinste Kripke-Struktur für Pfade, die $\neg\varphi$ erfüllen
  - Benötigt Fairness-Bedingungen (Fair-CTL)
- Bilde Produkt-Kripke-Struktur $K \times T_{\neg\varphi}$
  - $K \times T_{\neg\varphi}$ hat keinen Pfad $K \times T_{\neg\varphi} \models \neg EG\ true$

Motivierendes Beispiel

- Gilt ¬(p U q) in der folgenden Kripke-Struktur ?



Idee: Vergleiche mit „allgemeinster" Kripke-Struktur, die ¬¬(pUq) = (pUq) erfüllt

```
1  MODULE main
2  VAR
3    p : boolean;
4    q: boolean;
5  ASSIGN
6    next(p) := case
7      !p : 0;
8       q : 1;
9       1 : {0,1};
10   esac;
11   next(q) := case
12     p & next(p) : !q;
13     !p : 1;
14      1 : {0,1};
15   esac;
16
17 LTLSPEC
18    !(p U q)
```

```
Output Window
-> State: 1.1 <-
  p = 1
  q = 1
-> Input: 1.2 <-
-> State: 1.2 <-
  q = 0
-> Input: 1.3 <-
-> State: 1.3 <-
  p = 0
-> Input: 1.4 <-
-- Loop starts here
-> State: 1.4 <-
  q = 1
-> Input: 1.5 <-
-> State: 1.5 <-
```

Beachte, dass auch next() in der Bedingung eines case auftreten darf.

Tableau für Aussagenlogik

- Tableau für Formel φ
  - Systematische, vollständige Fallunterscheidung für Gültigkeit
    - Konstruiere Tableau für $\neg\varphi$
    - Zeige, dass es unerfüllbar ist
- Nur Teilformeln von φ spielen eine Rolle
  - $\vee$, $\exists$ führen zur Fallunterscheidung
    - Aufspaltung des Tableaux
  - $\wedge$, $\forall$ führen zur Akkumulation der Bestandteile
    - Pfadverlängerung
  - $\neg$ werden nach innen gebracht, $\neg\neg$ werden entfernt
    - $\neg$ stehen nur vor atomaren Symbolen
- Tableaus für Aussagenlogik:
  - Bäume mit Teilformeln an Knoten
  - φ unerfüllbar $\Leftrightarrow$ Jeder Ast widersprüchlich

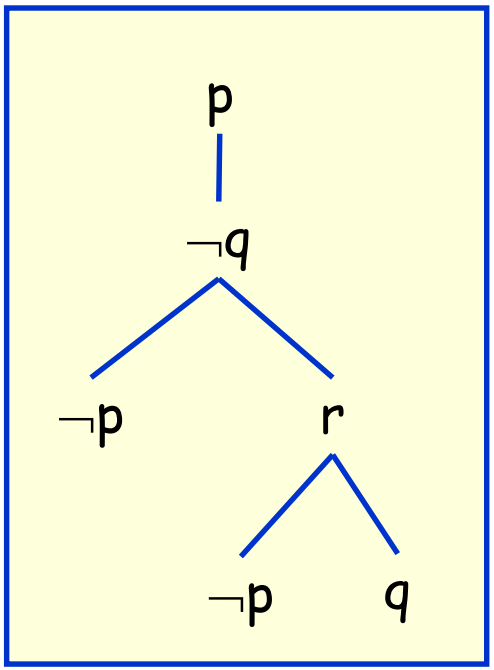


Tableau für

$p \wedge \neg q \wedge (\neg p \vee (r \wedge (\neg p \vee q)))$

Alle Pfade geschlossen

Tableau für LTL-Formel $\varphi = p\ U\ q$

- Eine Menge von M von Teilformeln von φ heißt konsistent, wenn es eine Spur gibt, die alle Formeln aus M erfüllt.
  - konsistent z.B.
    - $\{p,q,\varphi\}$, $\{p,\neg q,\varphi\}$, $\{\neg p,\neg q,\neg\varphi\}$
  - nicht konsistent z.B.:
    - $\{\neg p,\neg q,\varphi\}$, $\{p,q,\neg\varphi\}$, $\{\neg p,q,\neg\varphi\}$
- Zustände des Tableaus: maximale konsistente Mengen von Teilformeln von φ
  - $\{p,q,\varphi\}$, $\{p,\neg q,\varphi\}$, $\{\neg p,q,\varphi\}$, $\{p,\neg q,\neg\varphi\}$, $\{\neg p,\neg q,\neg\varphi\}$
- Übergang heißt konsistent, wenn es eine Spur gibt, die diesen Übergang realisiert
  - Konsistente Übergänge z.B.:
    - $\{p,\neg q,\varphi\} \rightarrow \{\neg p, q,\varphi\}$
    - $\{p,\neg q,\varphi\} \rightarrow \{p, \neg q,\varphi\}$
  - Nicht konsistent z.B.
    - $\{p,\neg q,\varphi\} \rightarrow \{\neg\varphi\}$
    - $\{p, \neg q,\neg\varphi\} \rightarrow \{\neg p, q,\varphi\}$
- Anfangszustände:
  - Alle, die φ enthalten

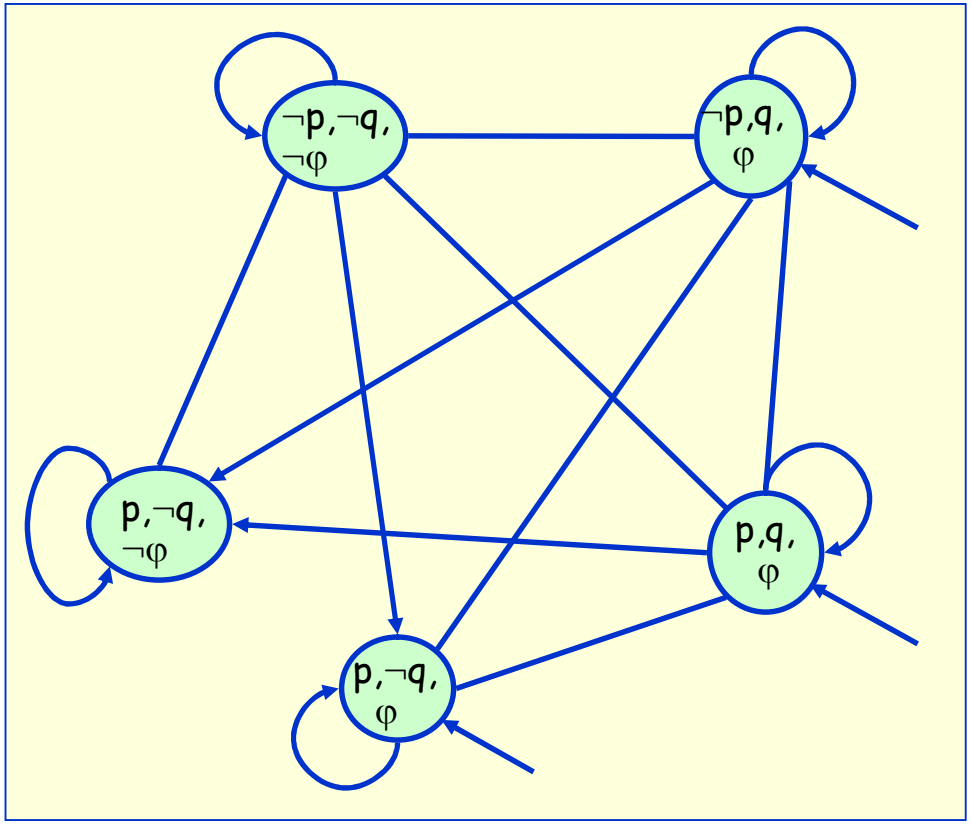
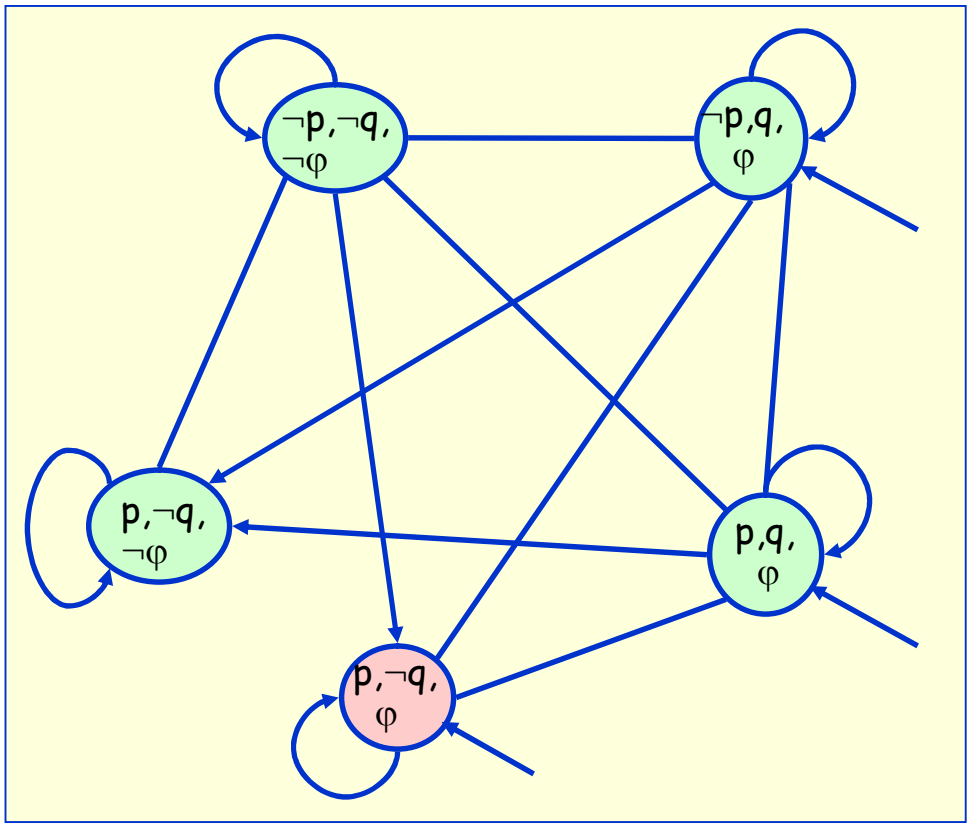


Tableau für $\varphi = p\ U\ q$

- Jede Spur, die φ erfüllt, wird von dem Tableau akzeptiert
- Leider aber auch die Spur
  - $\{p, \neg q, \varphi\}, \{p, \neg q, \varphi\}, \{p, \neg q, \varphi\}, \{p, \neg q, \varphi\}, \ldots$
  - sie löst ihr „Versprechen" nicht ein, irgendwann q zu erfüllen.
- Fairness Bedingung:
  - $GF\ (\varphi \Rightarrow q)$

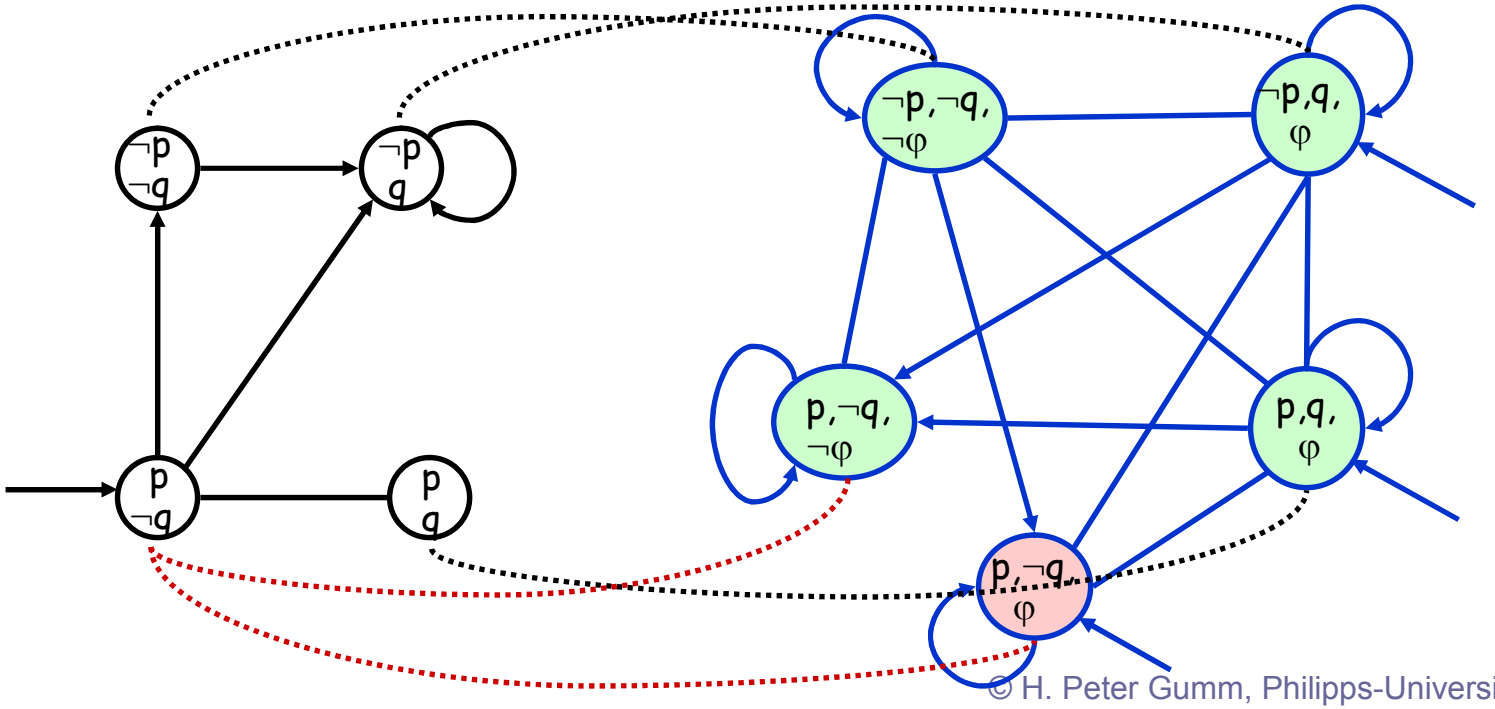
Begründung: Angenommen, $\sigma \not\models GF(\varphi \Rightarrow q)$.
Dann gilt $\sigma \models FG\ \neg(\varphi \Rightarrow q)$, also $\sigma \models FG\ (\varphi \wedge \neg q)$.

Genau solche Pfade wollen wir aber ausschließen.



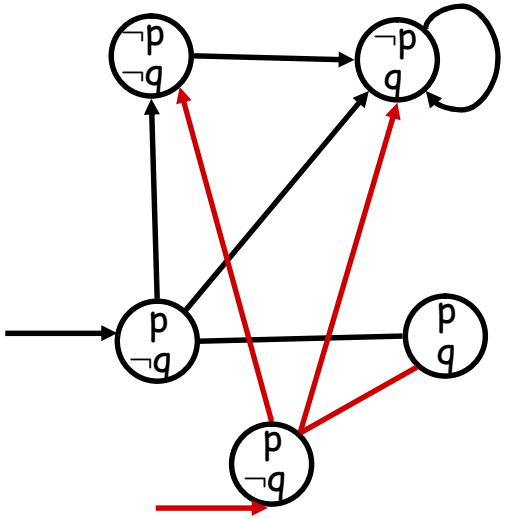
Vergleiche Tableau mit Kripke-Struktur

- Idee: Laufe „gleichzeitig" durch Kripke-Struktur und entsprechende Zustände im Tableau
  - „entsprechend": die gleichen Labels
  - „gleichzeitig": Transitionen komponentenweise
  - technisch: entsprechende Paare aus dem Kartesischen Produkt
- Im Beispiel führt dies nur zur „Aufspaltung" von {p,¬ q} in ({p,¬ q},{p,¬ q,¬φ}) und ({p,¬ q},{p,¬ q,φ})

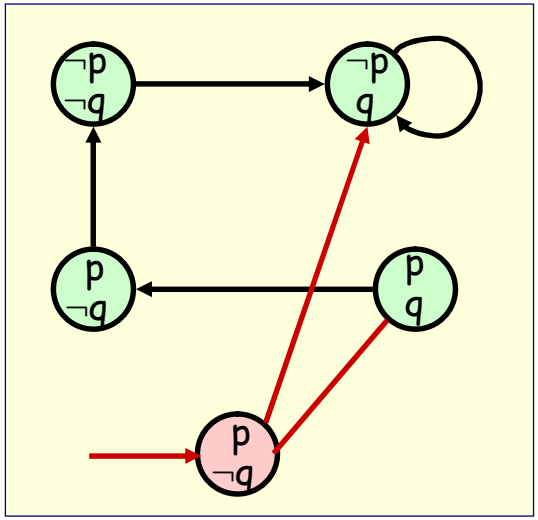


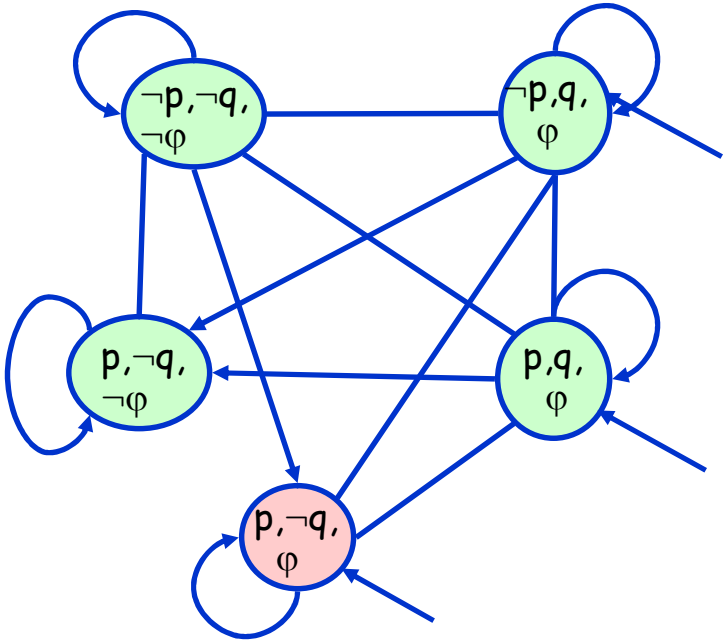
Vergleiche Tableau mit Kripke-Struktur

- Idee: Laufe „gleichzeitig" durch Kripke-Struktur und entsprechende Zustände im Tableau
  - entsprechend: die gleichen Labels
  - gleichzeitig: Transitionen komponentenweise
  - technisch: entsprechende Paare aus dem Kartesischen Produkt



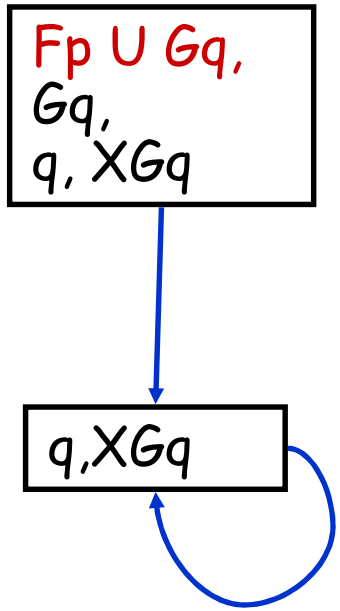
Produkt





Algorithmus: Tableaukonstruktion

- Tableau für Formel ψ aus LTL:
  - Zustände: Maximale konsistente Mengen von Teilformeln von ψ :
    - $\varphi \in s \quad :\Leftrightarrow \quad s \models \varphi$
  - Transitionen
    - Bestimmt durch Teilformeln $X\varphi$ von ψ :
    - $X\varphi \in s,\ s \rightarrow s' \Rightarrow s' \models \varphi$

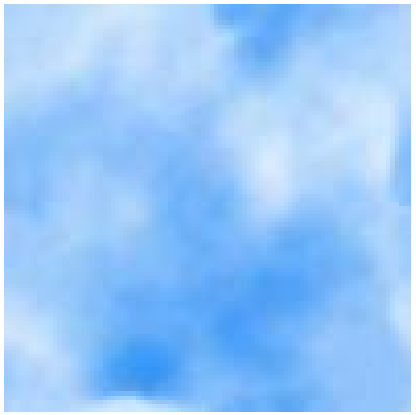


Hintergrund: Warum soll φ in s gelten?

- φ temporale Formel, K = (S,R,L) Kripke-Struktur, $s \in S$ mit $s \models \varphi$.
- Grund, warum φ in s gilt:
  - $\varphi = p$: weil $p \in L(s)$
  - $\varphi = \varphi_1 \wedge \varphi_2$: weil $s \models \varphi_1$ und $s \models \varphi_2$
  - $\varphi = \varphi_1 \vee \varphi_2$: weil $s \models \varphi_1$ oder $s \models \varphi_2$
  - $\varphi = \neg \varphi_1$: weil $s \models \varphi_1$ nicht gilt.
  - $\varphi = X \varphi_1$: weil in jedem Folgezustand $s' \models \varphi_1$.
  - $\varphi = \varphi_1 \, U \, \varphi_2$: weil $s \models \varphi_2 \vee (\varphi_1 \wedge X(\varphi_1 \, U \, \varphi_2))$,

    d.h. $s \models \varphi_2$ oder ($s \models \varphi_1$ und $s \models X(\varphi_1 \, U \, \varphi_2)$)
  - $\varphi = \varphi_1 \, R \, \varphi_2$: weil $s \models \varphi_2 \wedge (\varphi_1 \vee X(\varphi_1 \, R \, \varphi_2))$,

    d.h. weil $s \models \varphi_2$ und ($s \models \varphi_1$ oder $s \models X(\varphi_1 \, R \, \varphi_2)$)



- Kurz: Weil bestimmte „*Teilformeln*“ in s oder den Nachfolgern von s gelten.
- Was sind *Teilformeln* und wie sieht die Menge der Teilformeln aus, die in einem Zustand gelten

Erweiterte Definition von Teilformeln

$Sub(p) = p$, falls $p \in AP$

$Sub(\varphi_1 \wedge \varphi_2) = \{\varphi_1 \wedge \varphi_2\} \cup Sub(\varphi_1) \cup Sub(\varphi_2)$
$Sub(\varphi_1 \vee \varphi_2) = \{\varphi_1 \vee \varphi_2\} \cup Sub(\varphi_1) \cup Sub(\varphi_2)$
$Sub(\neg \varphi) = \{\neg \varphi\} \cup Sub(\varphi)$

$Sub(X \varphi) = \{X \varphi\} \cup Sub(\varphi)$

$Sub(\varphi \, U \, \psi) = \{\varphi \, U \, \psi, X(\varphi \, U \, \psi)\} \cup Sub(\varphi) \cup Sub(\psi)$
$Sub(\varphi \, R \, \psi) = \{\varphi \, R \, \psi, X(\varphi \, R \, \psi)\} \cup Sub(\varphi) \cup Sub(\psi)$

es folgt:

$Sub(G \varphi) = \{G\varphi, \varphi, X(G\varphi)\} \cup Sub(\varphi)$
$Sub(F \varphi) = \{F\varphi, \varphi, X(F\varphi)\} \cup Sub(\varphi)$

Aufgrund der FP-Gleichungen:

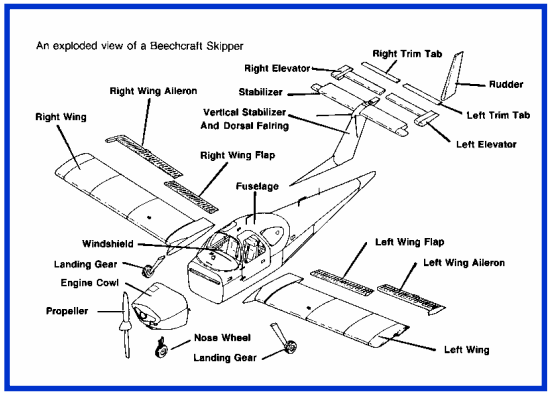
$\varphi \, U \, \psi = \psi \vee (\varphi \wedge X(\varphi \, U \, \psi))$
$\varphi \, R \, \psi = \psi \wedge (\varphi \vee X(\varphi \, R \, \psi))$

$G \varphi = (\varphi \wedge X(G \varphi))$
$F \varphi = (\varphi \vee X(F \varphi))$

Welche Teilformeln gelten in s

- φ gelte in einem Zustand s einer Kripke-Struktur
- welche Teilformeln von φ gelten dann auf jeden Fall auch in s ?
  - $\varphi = \varphi_1 \wedge \varphi_2$: φ_1 und φ_2
  - $\varphi = \varphi_1 \vee \varphi_2$: φ_1 oder φ_2
  - $\varphi = \neg\varphi_1$: φ_1 jedenfalls nicht
  - $\varphi = X\ \varphi_1$: keine direkte Teilformel
  - $\varphi = \varphi_1\ U\ \varphi_2$: φ_2 oder (φ_1 und $X(\varphi_1\ U\ \varphi_2)$)
  - $\varphi = \varphi_1\ R\ \varphi_2$: φ_2 und (φ_1 oder $X(\varphi_1\ R\ \varphi_2)$)
  - $\varphi = G\ \varphi_1$: φ_1 und $X\ G\ \varphi_1$
  - $\varphi = F\ \varphi_1$: φ_1 oder $X\ F\ \varphi_1$

Konsistente Mengen von Teilformeln



- $s \subseteq \mathrm{Sub}(\varphi)$ heißt konsistent, falls
  - $\varphi_1 \wedge \varphi_2 \in s \Rightarrow \varphi_1 \in s$ und $\varphi_2 \in s$
  - $\varphi_1 \vee \varphi_2 \in s \Rightarrow \varphi_1 \in s$ oder $\varphi_2 \in s$
  - $\neg\varphi_1 \in s \Rightarrow \varphi_1 \notin s$
  - $X\varphi_1 \in s \Rightarrow$ keine Bedingung
  - $\varphi_1 \,U\, \varphi_2 \in s \Rightarrow \varphi_2 \in s$ oder ($\varphi_1 \in s$ und $X(\varphi_1 \,U\, \varphi_2) \in s$)
  - $\varphi_1 \,R\, \varphi_2 \in s \Rightarrow \varphi_2 \in s$ und ($\varphi_1 \in s$ oder $X(\varphi_1 \,R\, \varphi_2)) \in s$)
- Es folgt
  - $G\,\varphi \in s \Rightarrow \varphi \in s$ und $X\,G\varphi \in s$
  - $F\,\varphi \in s \Rightarrow \varphi \in s$ oder $X\,F\varphi \in s$
- Ist s ein Zustand und φ eine LTL-Formel, dann ist

  $T(s,\varphi) := \{\psi \in \mathrm{Sub}(\varphi) \mid s \models \psi\}$

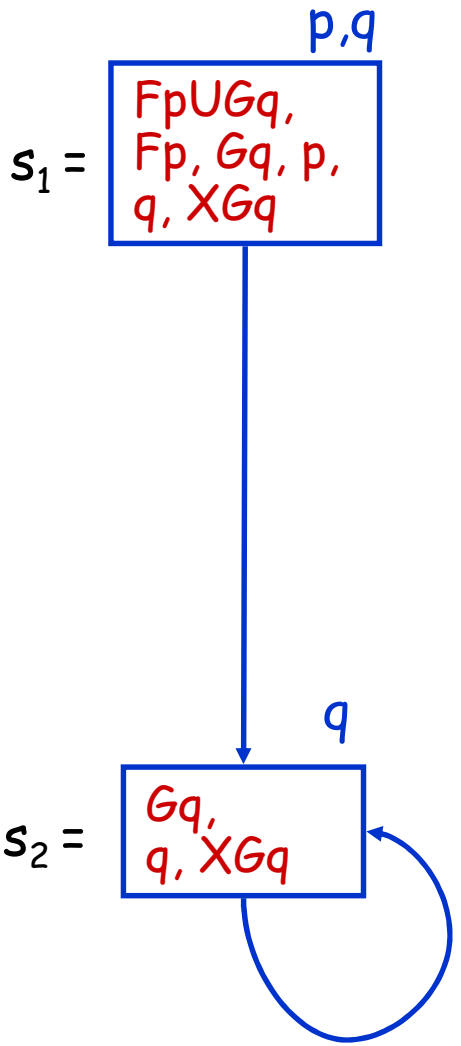
  eine maximale konsistente Menge von Teilformeln von φ
- Idee:
  - Drehe das um ... Zustände sind maximale konsistente Teilmengen von $\mathrm{Sub}(\varphi)$

Tableau-Konstruktion von Kφ

- Kripke-Struktur $K\varphi := (S, S_0, \rightarrow, L)$
- S Menge aller konsistenten Teilmengen von Sub(φ)
- $s \in S_0$ gdw. $\varphi \in s$
- $s \rightarrow t$ gdw. für alle $X\psi \in Sub(\varphi)$:

$$X\psi \in s \Rightarrow \psi \in t$$

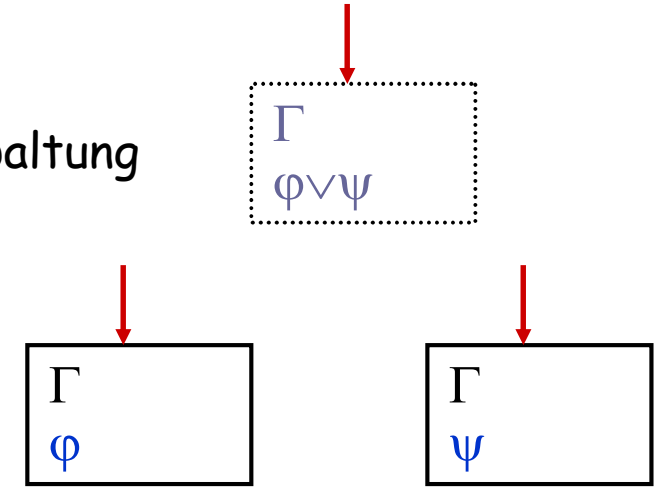
- L(s) Menge aller atomaren Formeln in s

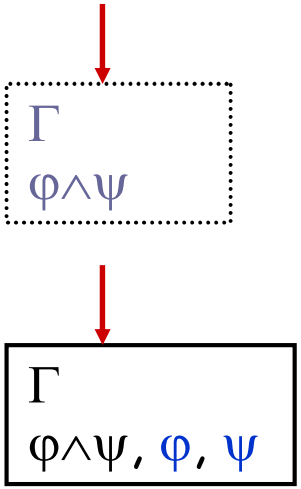


Konstruktion der konsistenten Teilmengen

Γ stehe für eine Menge bereits vorhandener Teilformeln

Aufspaltung

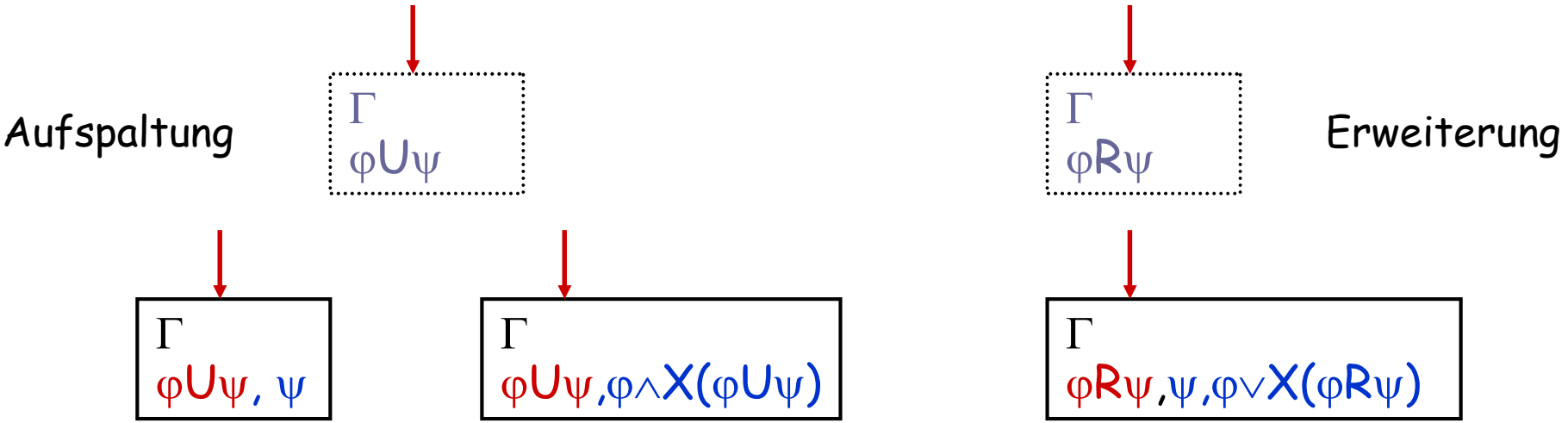




Erweiterung

Konstruktion der konsistenten Teilmengen

Γ stehe für eine Menge bereits vorhandener Teilformeln

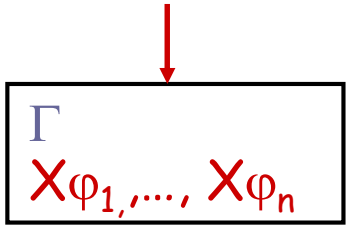


wegen Fixpunktgleichungen

$$\varphi U \psi = \psi \vee (\varphi \wedge X(\varphi U \psi))$$

$$\varphi R \psi = \psi \wedge (\varphi \vee X(\varphi R \psi))$$

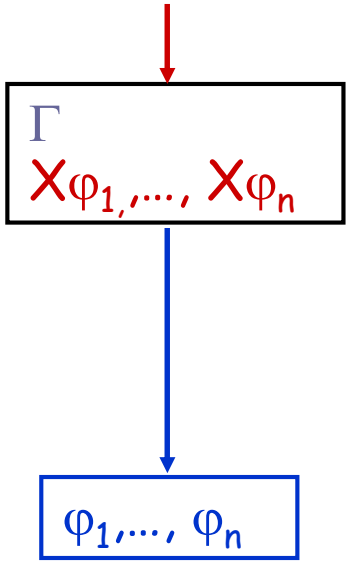
Zustandsübergänge



Falls keine der
vorigen Regeln
mehr anwendbar
ist eine konsistente
Menge erreicht

Zustandsübergänge

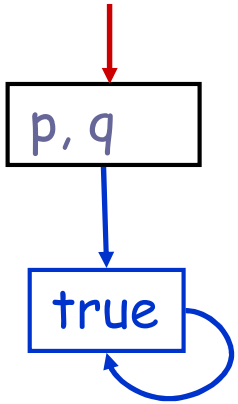
Falls keine der vorigen Regeln mehr anwendbar



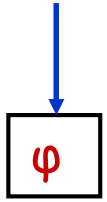
Behalte alten Zustand bei. Erzeuge neuen Zustand und Transition.

Nichttemporale Knoten

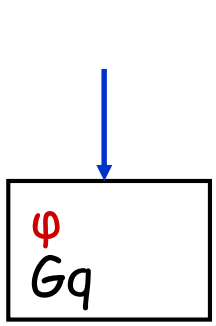
- Ein Knoten mit rein aussagenlogischen Formeln ist uninteressant
  - Keine weitere Verpflichtung für den Lauf
- Transition zu Default-Knoten { }

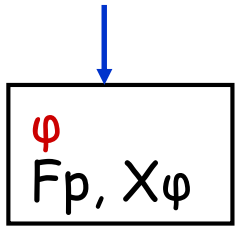


Beispiel: φ = (Fp U Gq)

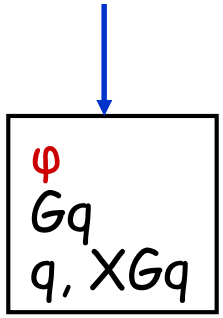


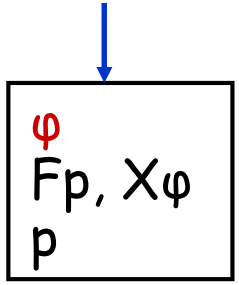
Beispiel: φ = (Fp U Gq)

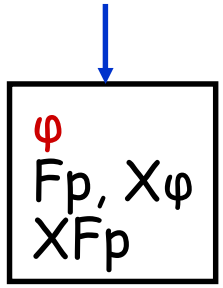




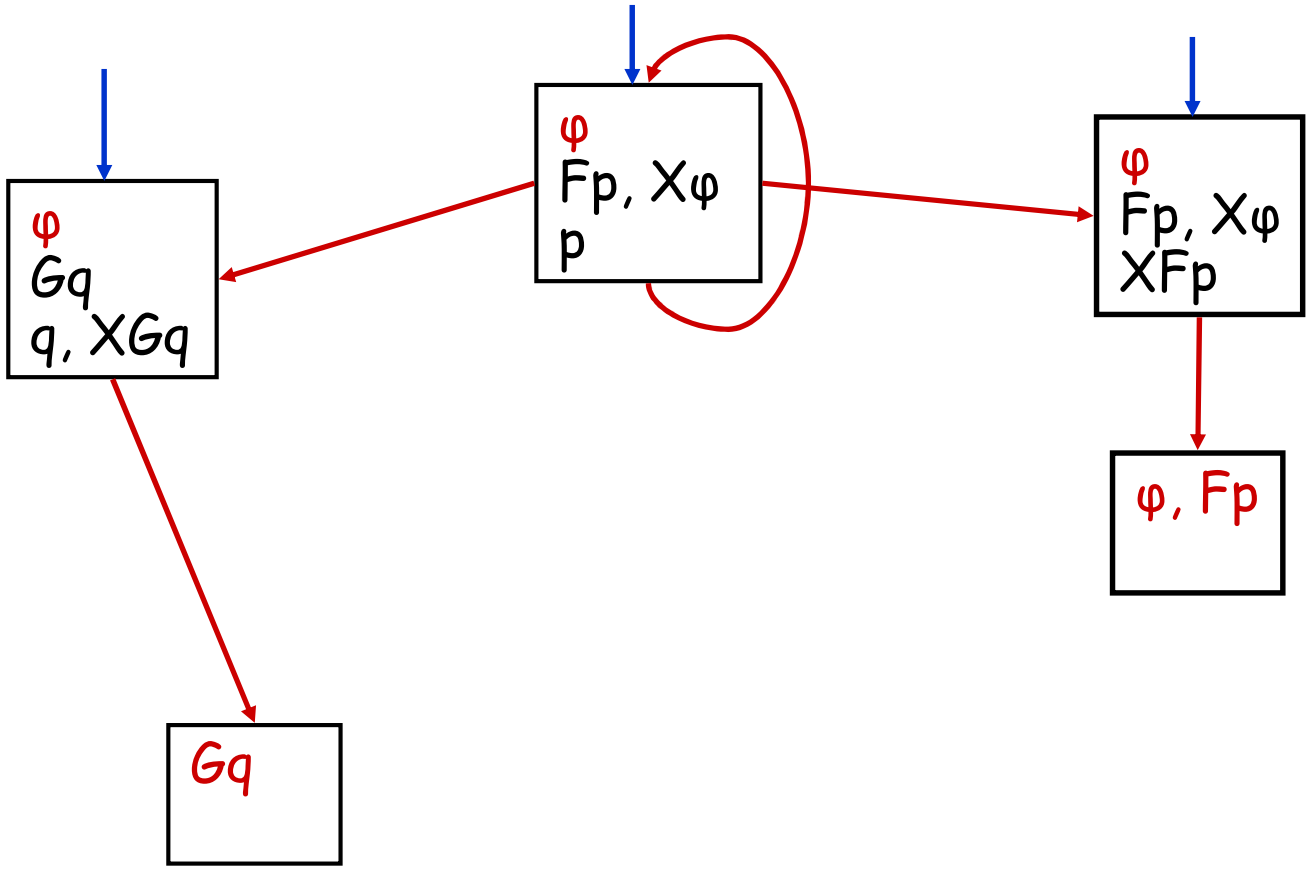
Beispiel: φ = (Fp U Gq)



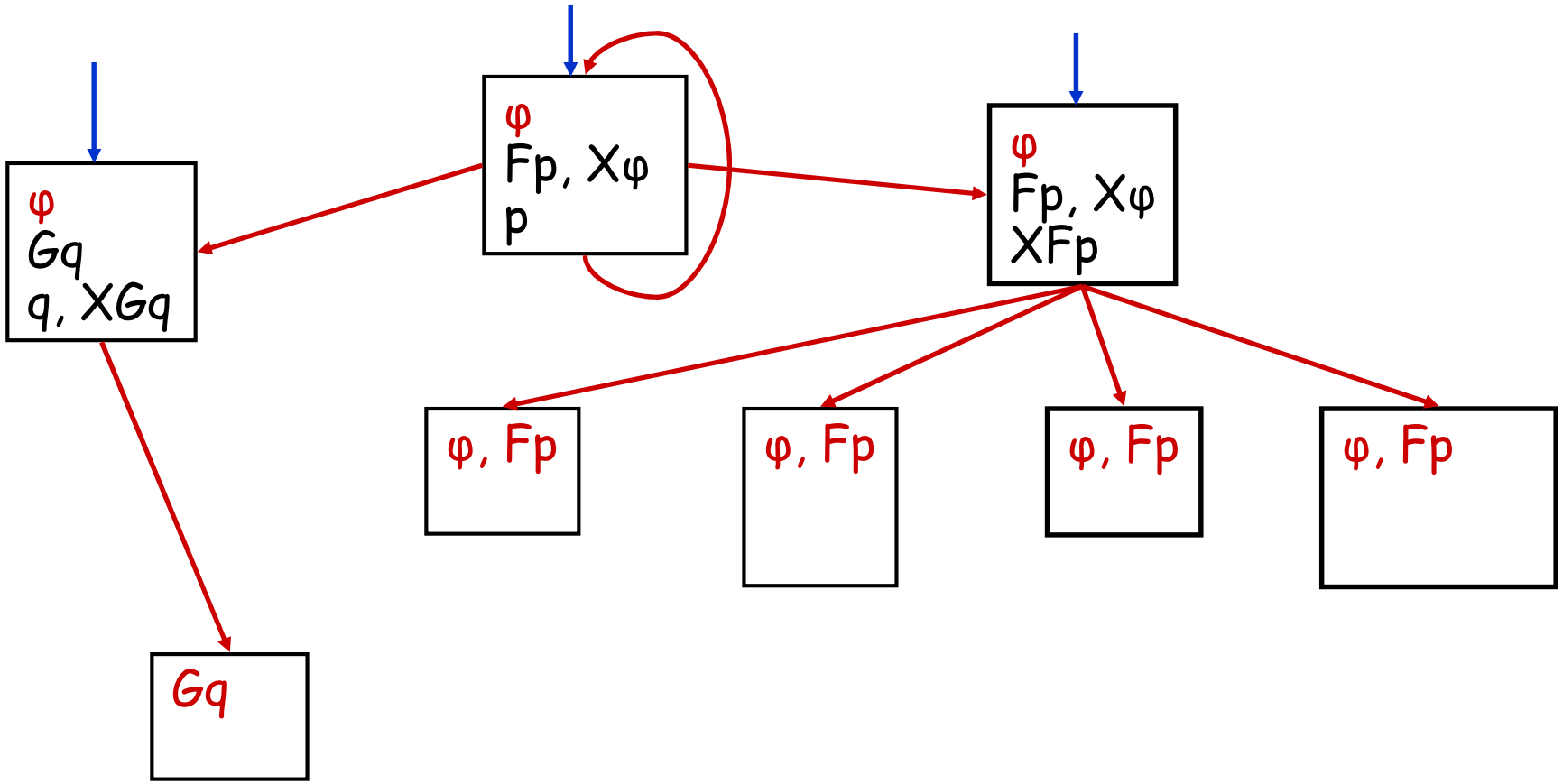




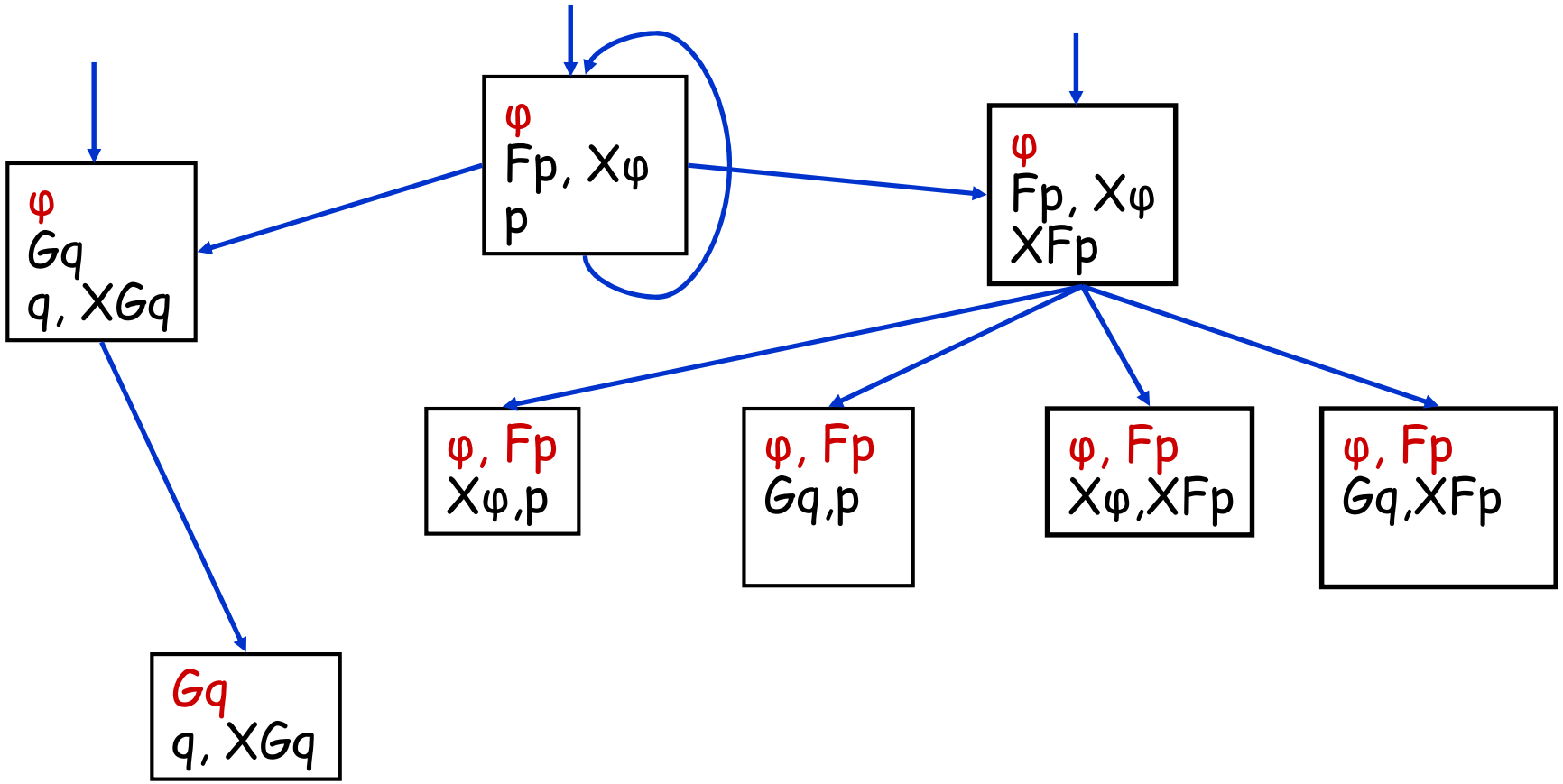
Beispiel: φ = (Fp U Gq)



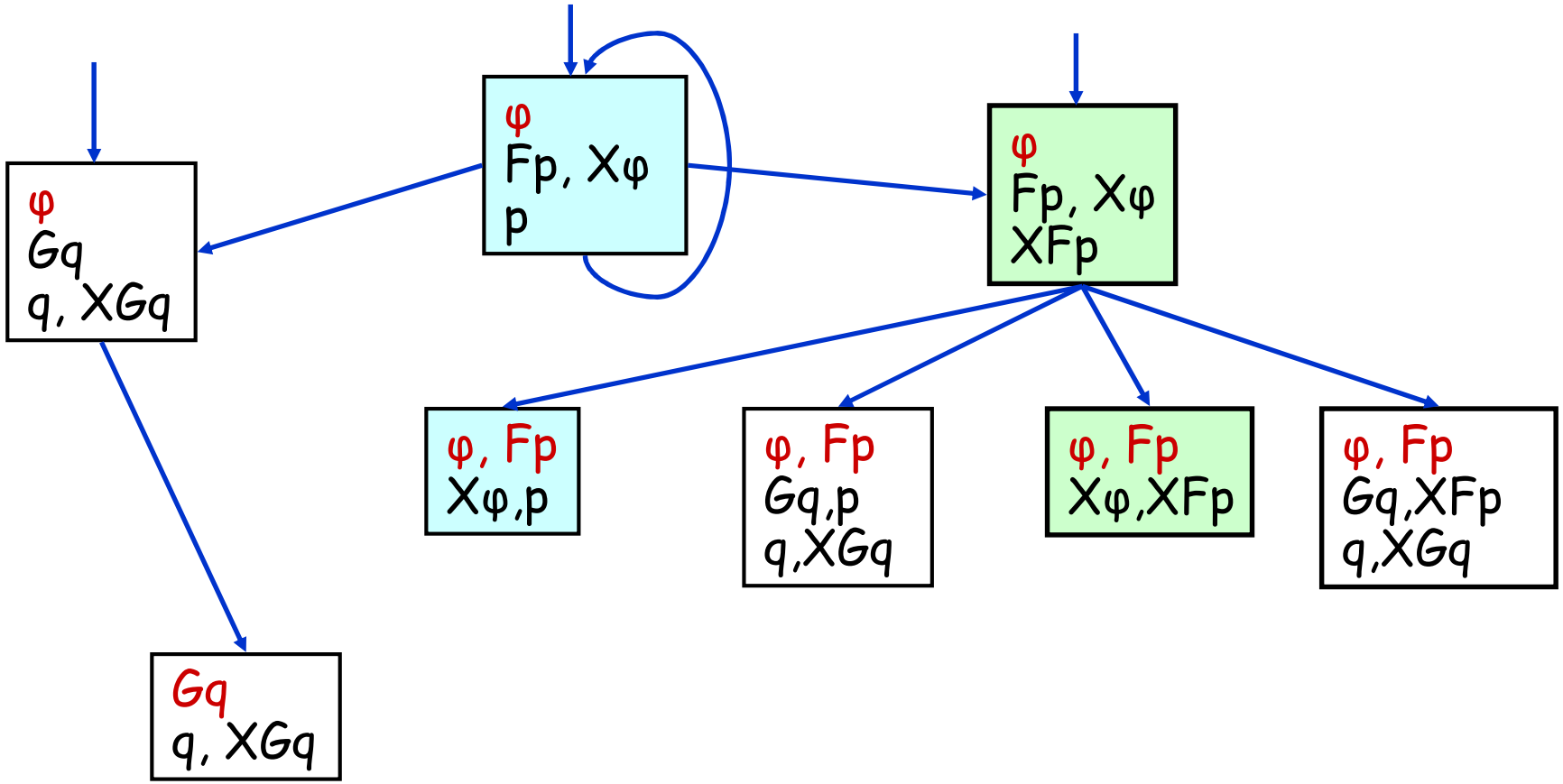
Beispiel: φ = (Fp U Gq)



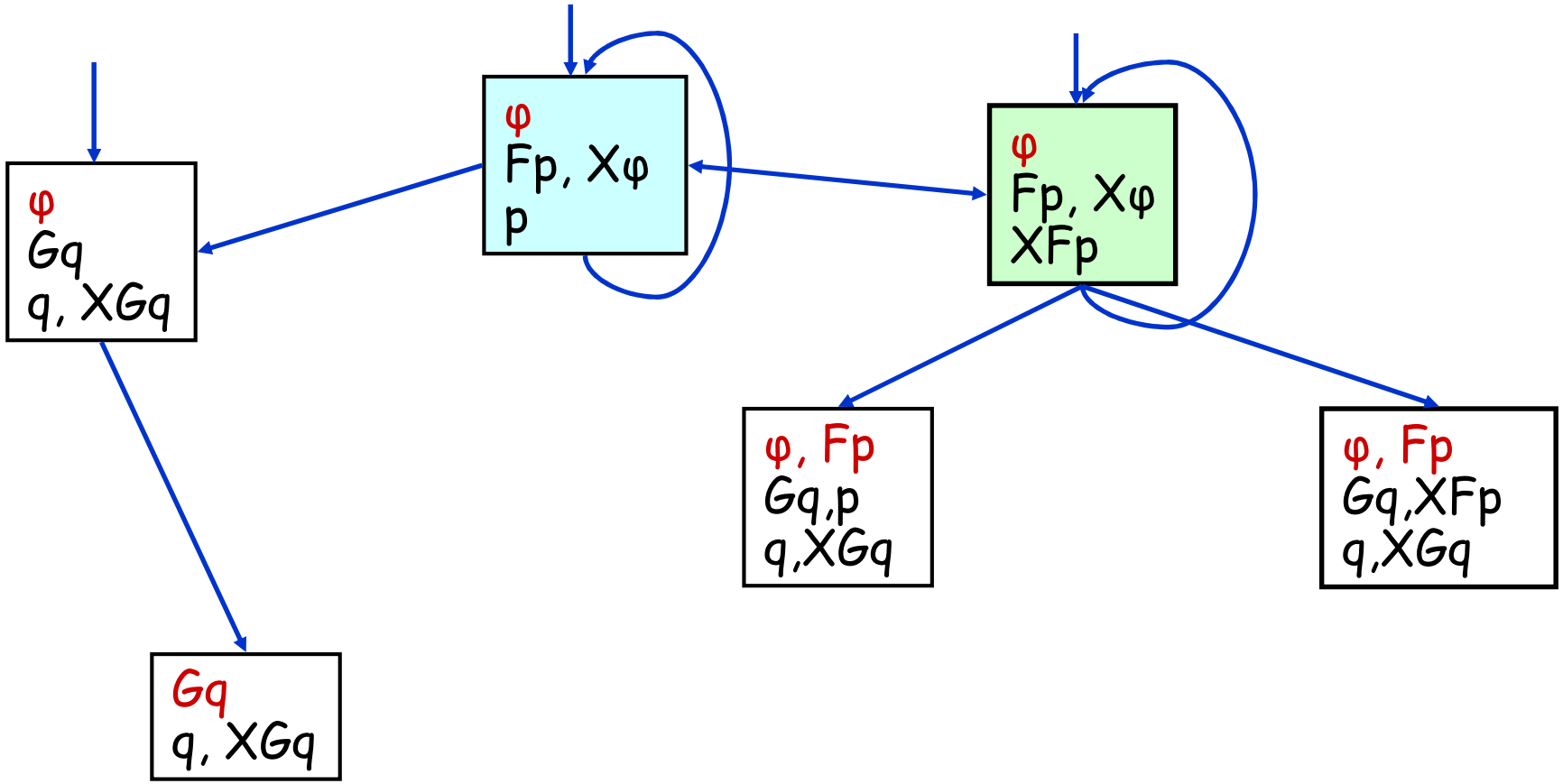
Beispiel: φ = (Fp U Gq)



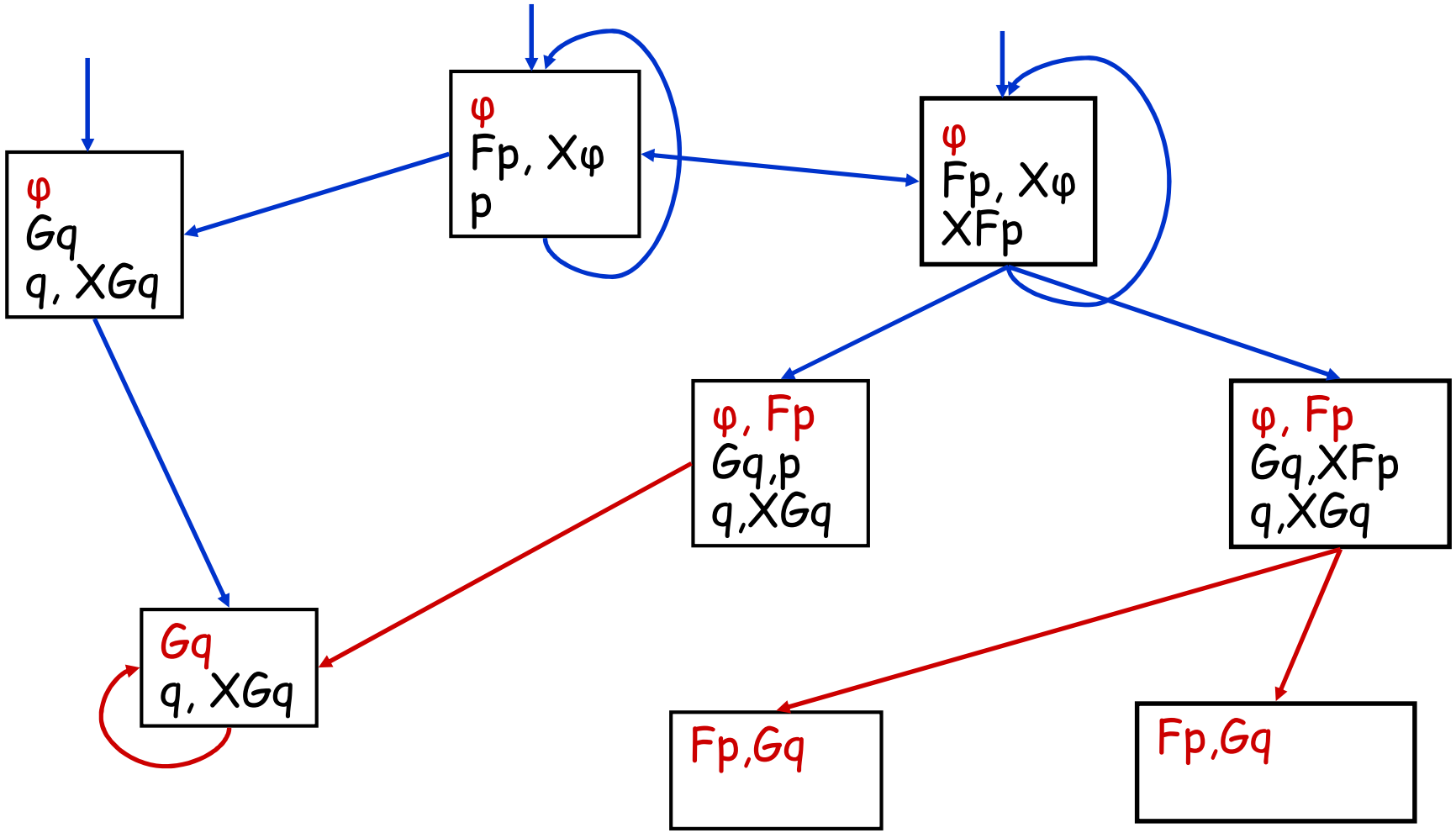
Beispiel: φ = (Fp U Gq)



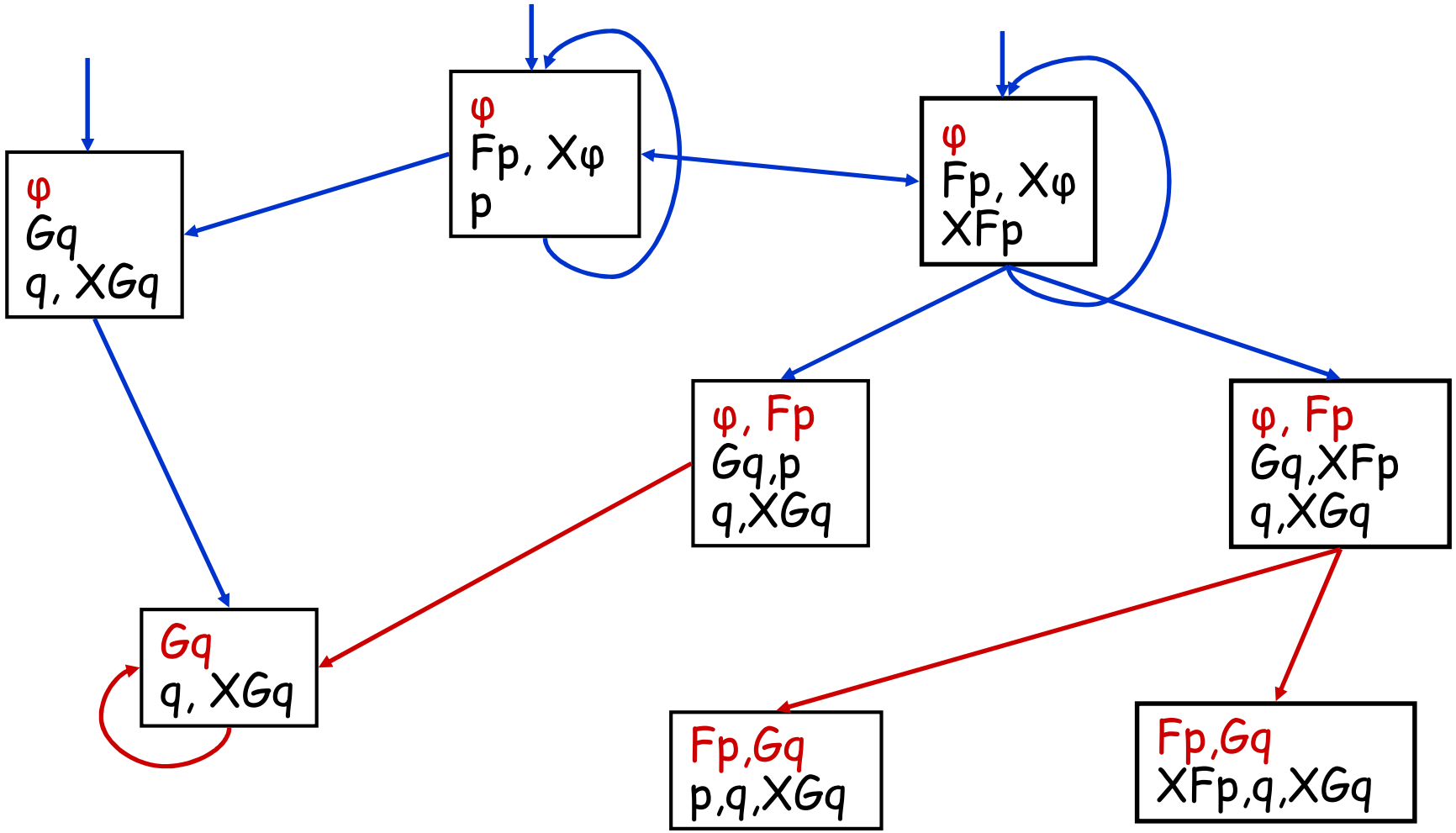
Beispiel: φ = (Fp U Gq)



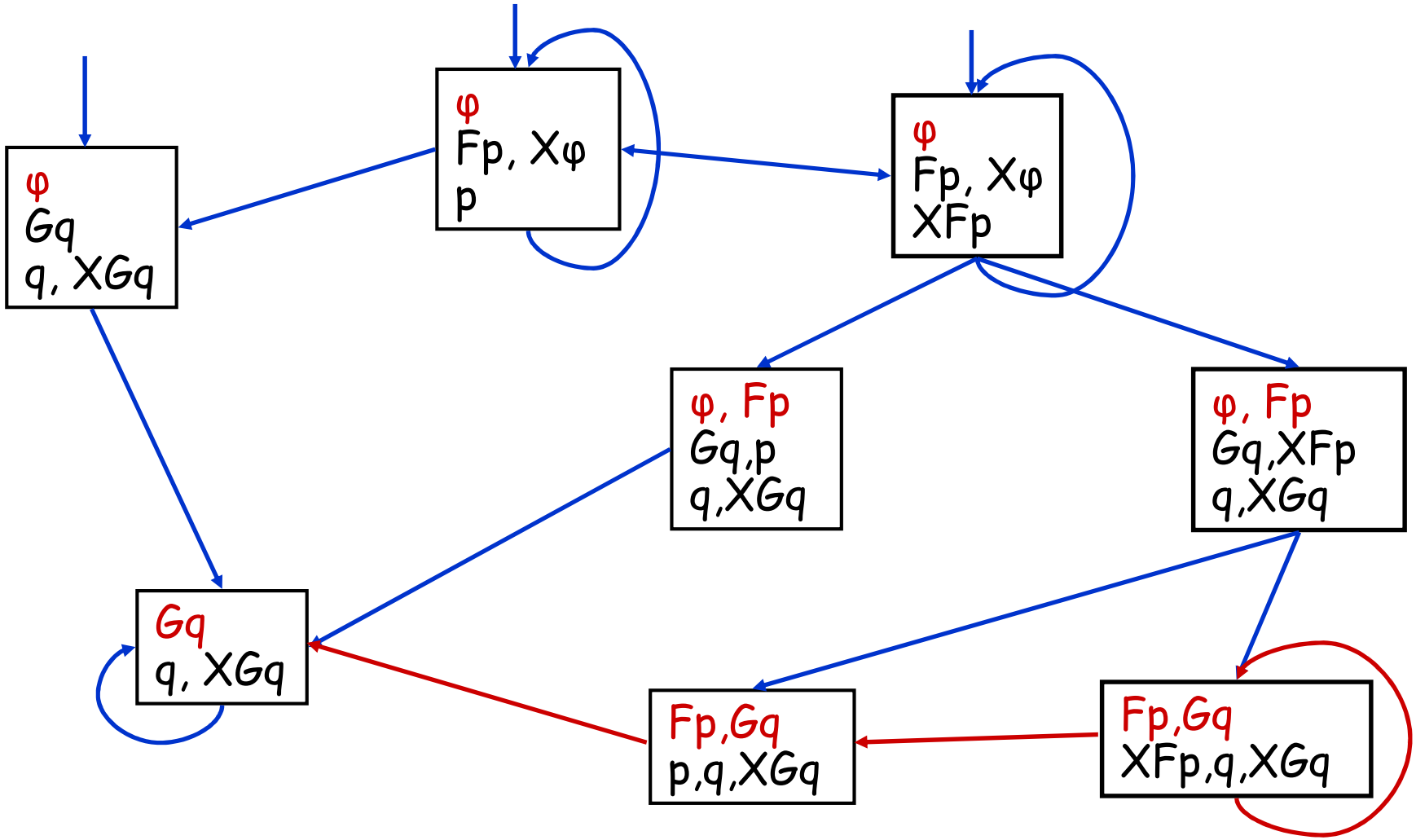
Beispiel: φ = (Fp U Gq)



Beispiel: φ = (Fp U Gq)



Beispiel: φ = (Fp U Gq)



Akzeptierung – in Fair-CTL

- Jede Formel $p\ U\ q \in s$ ist eine Obligation
- Entweder
  - $q \in s$: Obligation erfüllt
  - $X(p\ U\ q) \in s$: Obligation vertagt
- Ausgeschlossen werden müssen die Pfade, die
  - eine Obligation dauernd vertagen,
  - d.h. die $FG\ ((p\ U\ q) \wedge \neg q)$ erfüllen.
- Akzeptiert werden müssen alle Pfade, die für jede Teilformel der Form pUq
  - $GF\ (\neg(p\ U\ q) \vee q)$ erfüllen
- Das ist in Fair-CTL beschreibbar:
  - **FAIRNESS** $\neg(p\ U\ q) \vee q$
  - für jede Teilformel der Form pUq.

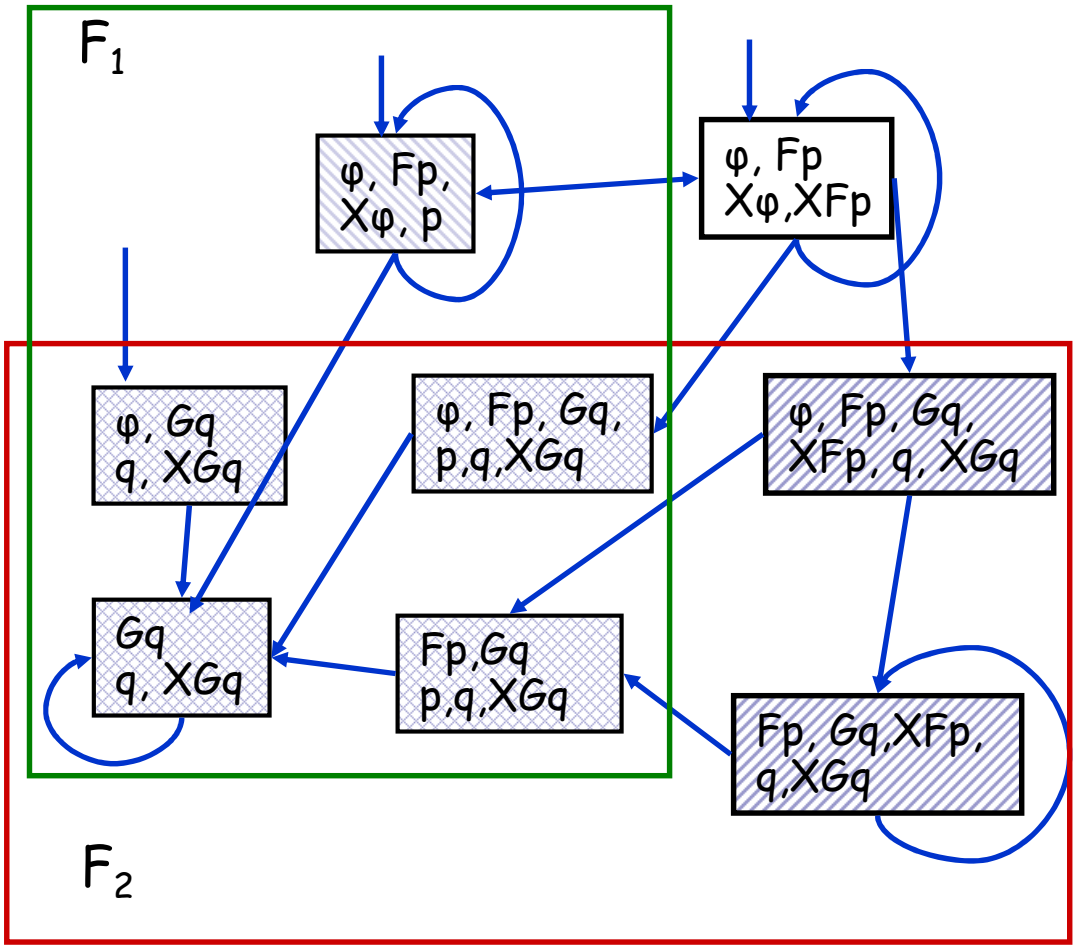
Akzeptanz

$F_1 = \{ s \mid Fp \in s \Rightarrow \quad p \in s \}$

$F_2 = \{ s \mid (Fp\ U\ Gq) \in s \Rightarrow Gq \in s \}$

- Kein Versprechen darf auf Dauer uneingelöst bleiben
  - Versprechen : Formel der Art $p\ U\ q$
  - Einlösung : Zustand mit q
- Nicht erlaubt:
  - Pfad π auf dem $p\ U\ q$, ab dort aber nie q gilt, denn nach Konstruktion müsste dann auch :
    - $\pi \models FG\ (\ X\ (p\ U\ q)\ \wedge\ \neg q\ )$, somit
    - $\pi \models FG\ (\ (p\ U\ q)\ \wedge \neg q)$ gelten.
  - Daher werden nur Pfade τ zugelassen mit

    $\tau \models GF\ ((p\ U\ q) \Rightarrow q)$, also
  - Akzeptanzmengen:
    - $F = \{ s \mid s \models p\ U\ q \Rightarrow q \} = [\ pUq \Rightarrow q\ ]$
  - Interpretation als Kripke-Struktur
    - FAIRNESS $(p\ U\ q) \Rightarrow q$
  - Argument gilt für jede Teilformel der Form $p\ U\ q$.

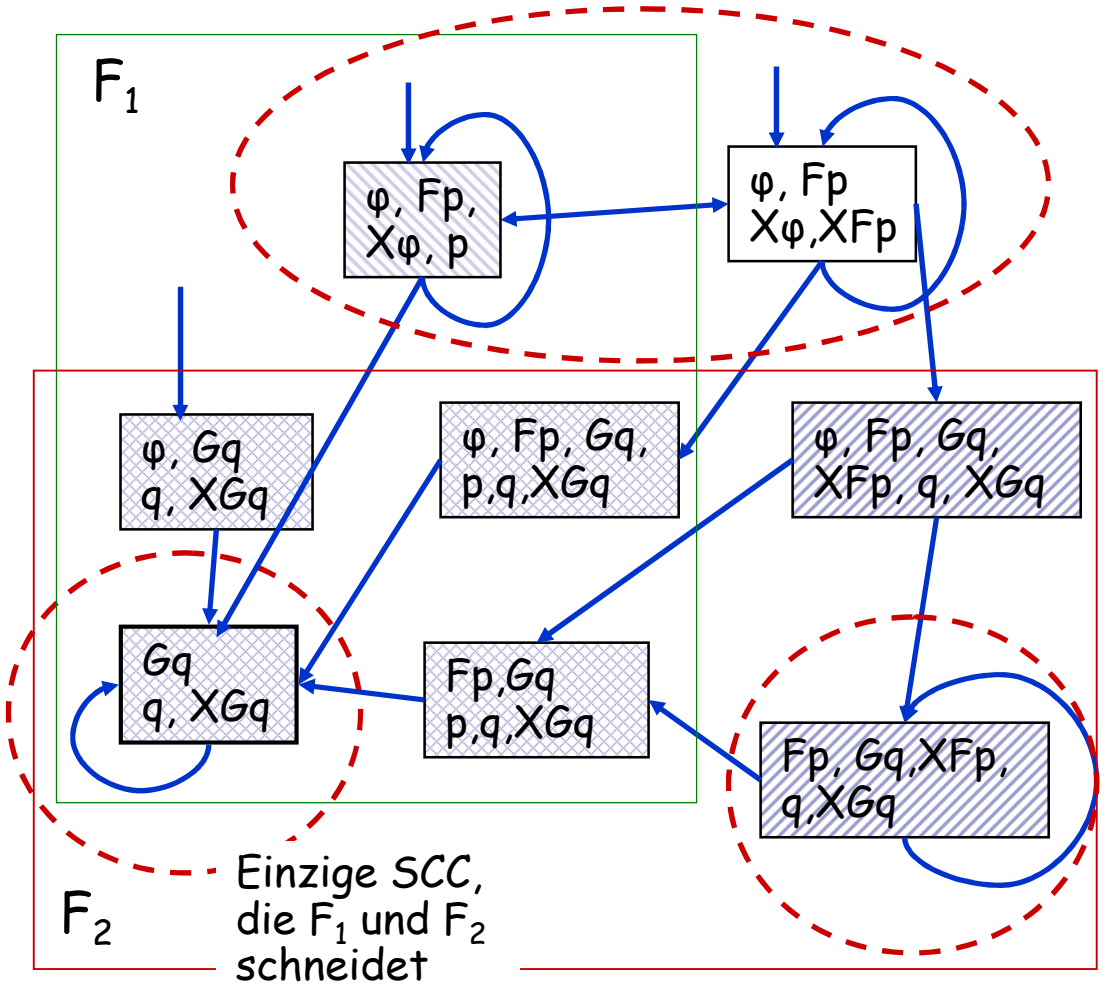


SCCs

$F_1 = \{ s \mid Fp \in s \Rightarrow p \in s \}$

$F_2 = \{ s \mid (Fp\ U\ Gq) \in s \Rightarrow Gq \in s \}$

- Jede unendliche Folge in einer endlichen Kripke-Struktur bleibt ab irgendeinem Zeitpunkt in einer SCC.
- Damit sie alle Akzeptanzbedingungen erfüllt, muss diese SCC alle Akzeptanzmengen schneiden.
- $\Rightarrow$ Bestimme alle SCCs
  - Behalte nur die, welche alle Akzeptanzbedingungen schneiden.
  - Die Vereinigung dieser SCCs ist die Akzeptanzmenge des Büchi-Automaten

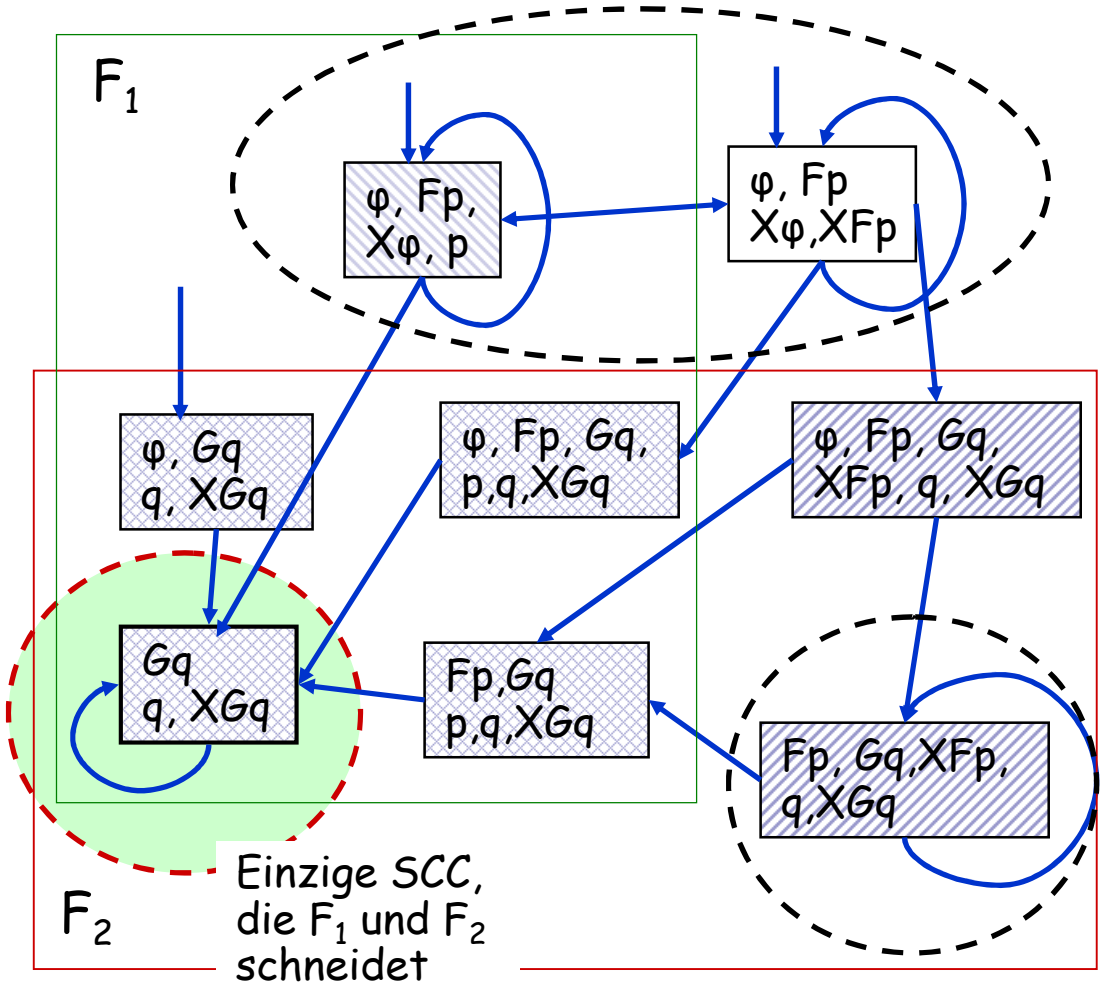


SCCs

$F_1 = \{ s \mid Fp \in s \Rightarrow p \in s \}$

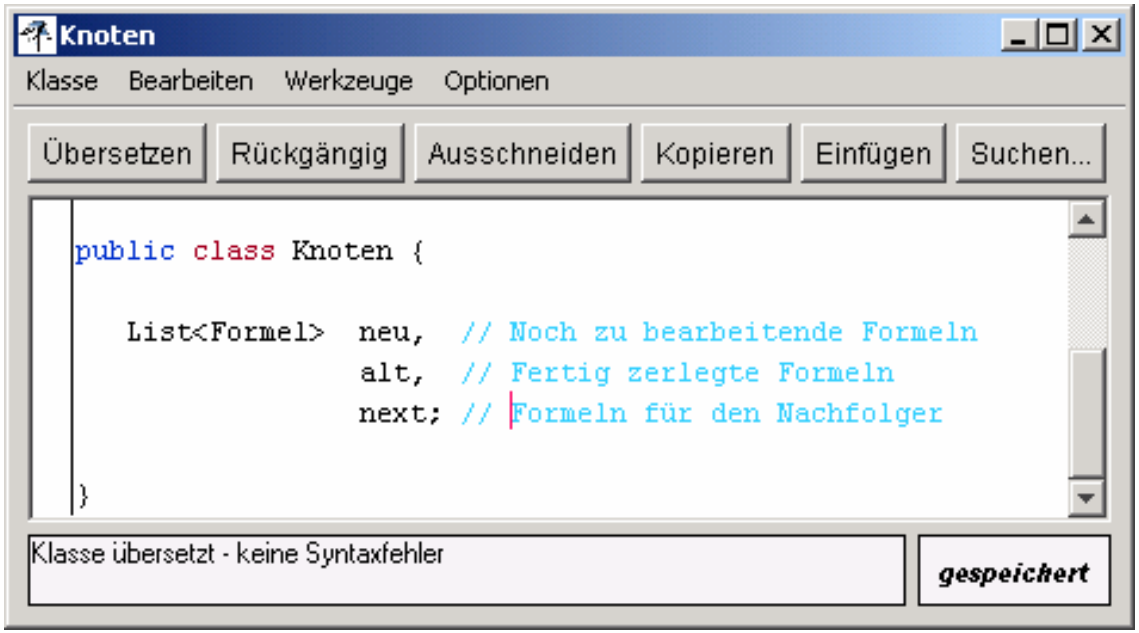
$F_2 = \{ s \mid (Fp\ U\ Gq) \in s \Rightarrow Gq \in s \}$

- Jede unendliche Folge in einer endlichen Kripke-Struktur bleibt ab irgendeinem Zeitpunkt in einer SCC.
- Damit sie alle Akzeptanzbedingungen erfüllt, muss diese SCC alle Akzeptanzmengen schneiden.
- $\Rightarrow$ Bestimme alle SCCs
  - Behalte nur die, welche alle Akzeptanzbedingungen schneiden.
  - Die Vereinigung dieser SCCs ist die Akzeptanzmenge des Büchi-Automaten



Implementierung

- Knoten haben drei Listen von Teilformeln
  - neu : zu expandierende Formeln
  - alt : fertig zerlegte Formeln
  - next : Formeln der Form $X\varphi$
- Ein Expandierungsschritt
  - Entfernt eine Formel aus neu
  - Generiert neue Formeln, die nach neu, alt und next verteilt werden
  - Eventuell eine Kopie des aktuellen Knotens
  - Bis neu leer ist.



α-Formeln und β-Formeln

- α-Formeln
  - Konjunktionen
    - $\varphi \wedge \psi$
  - ReleaseFormeln
    - $\varphi \; R \; \psi = \psi \wedge (\varphi \vee X(\varphi \; R \; \psi))$
  - Spezialfall:
    - $G \; \varphi = \varphi \wedge XG\varphi$

- β -Formeln
  - Disjunktionen
    - $\varphi \vee \psi$
  - Until-Formeln
    - $\varphi \; U \; \psi = \psi \vee (\varphi \wedge X(\varphi \; U \; \psi))$
  - Spezialfall
    - $F \; \varphi = \varphi \vee XF\varphi$

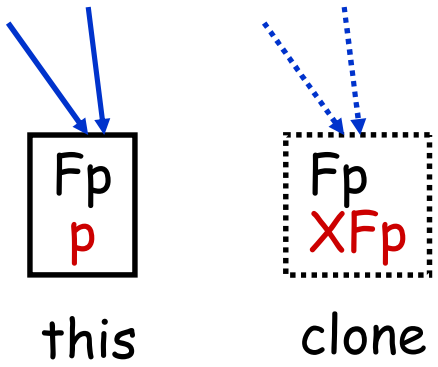
Expansion von α-Formel verändert nur neu, alt, next im aktuellen Knoten

Expansion von β-Formeln führt zu Aufspaltung (cloning) des Knotens
- in einem gilt die eine Alternative
- im clone die zweite

Expansionsschritte

- α-Formeln
  - Beispiel $G\varphi$
    - neu.remove($G\varphi$);
    - neu.add(φ);
    - next.add($G\varphi$);

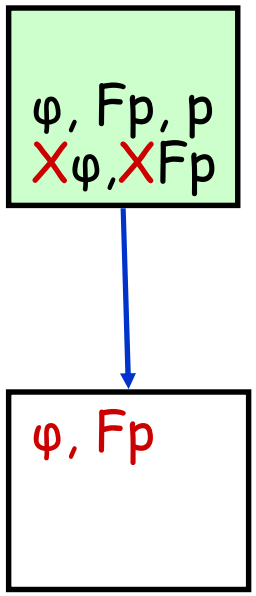
- β -Formeln
  - Beispiel φ U ψ
    - neu.remove(φ U ψ);
    - Knoten clone = clone();
    - Für jede Transition $s \rightarrow$ this füge Transition $s \rightarrow$ clone hinzu
    - this.neu.add(ψ);
    - clone.neu.add(φ);
    - clone.next.add(φ U ψ);
  - Spezialfall Fp:

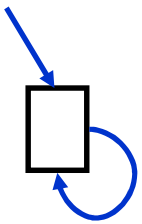


Literale in neu wandern sofort zu alt
X-Formeln aus neu wandern sofort zu next

Nachfolger

- Ein Knoten k ist voll expandiert, wenn
  - neu leer ist
- Dann:
  - generiere neuen Knoten k' mit
  - k'.neu = k.next
  - Transition: $k \rightarrow k'$
- Falls ein solcher Knoten schon besteht
  - Merge mit bestehendem Knoten
  - Vereinigung der Transitionen
- Spezialfall:
  - leerer Knoten





Produktkonstruktion

- $K=(S_K,R_K,L_K)$ sei die vorgelegte Kripke-Struktur, φ sei die LTL-Formel
- Konstruiere das Tableau $T=(S_T,R_T)$ für φ
- Konstruiere Produktsystem $P = (S_P,R_P,L_P)$ mit
  - $S_P = \{ (s,s') \in S_K \times S_T \mid \forall p \in AP.(p \in s' \Rightarrow p \in L_K(s)) \wedge ((\neg p) \in s' \Rightarrow p \notin L_K(s)) \}$
  - $(s,s') R_P (t,t') :\Leftrightarrow s R_K t,\ s' R_T t'$
- „LTL-Formel" $E\varphi$ in gilt in K $\Leftrightarrow$ Produktsystem erfüllt die CTL-Eigenschaft
  - SPEC EG true
  - und für jede Subformel $\alpha_i U \beta_i$
    - FAIRNESS $\neg(\alpha_i U \beta_i) \vee \beta_i$
  - bzw. für jede Subformel $F\alpha_i$
    - FAIRNESS $\neg F\alpha_i \vee \alpha_i$

Modifikation einer Konstruktion von

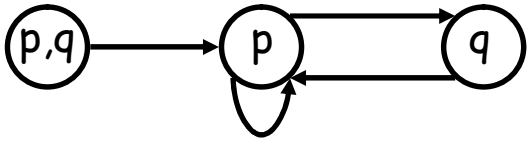
Clarke, Grumberg, Hamaguchi: *„Another Look at LTL-Model Checking"*

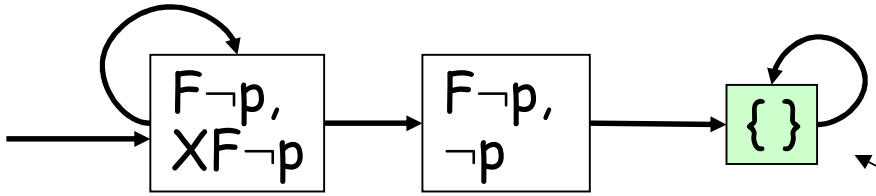
Zusammenfassung

- Gegeben
  - Kripke Struktur K
  - LTL-Formel φ
- Konstruiere
  - Kripke-Tableau $T_{\neg\varphi}$
  - Produkt-System $K \times T_{\neg\varphi}$
- Füge hinzu
  - FAIRNESS $\neg(p\ U\ q) \vee q$
    - Für jede Teilformel pUq von φ
- Überprüfe
  - EG true
    - wenn ja: φ nicht erfüllt in K
    - wenn nein: φ gilt in K

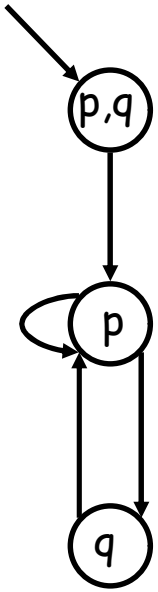
Beispiel

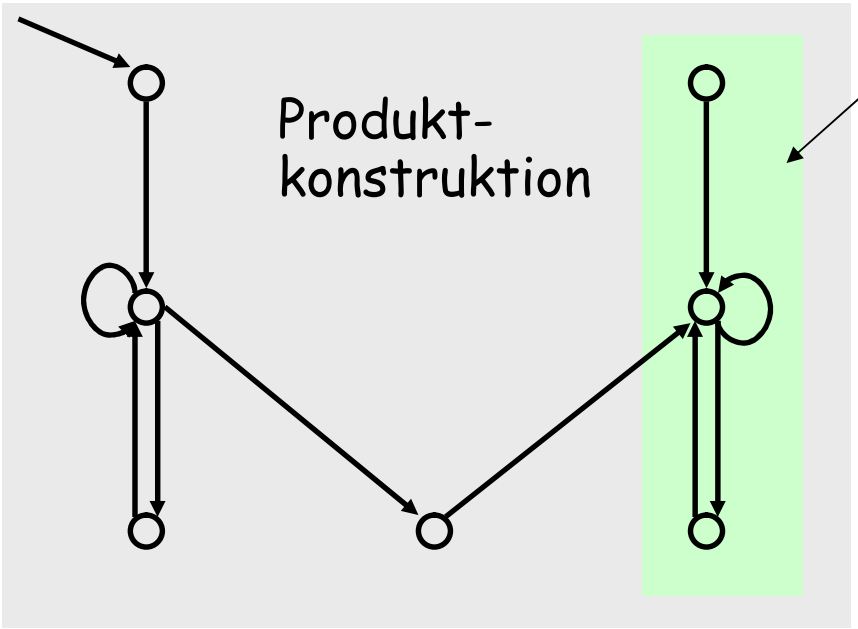
- Prüfe LTL-Formel $\varphi = G\ p$ in der Kripke-Struktur
- Negation $\neg\varphi = \neg\ G\ p$
- positive Normalform $\neg\varphi = F\neg\varphi$
- Tableau:





Fairnessmenge F

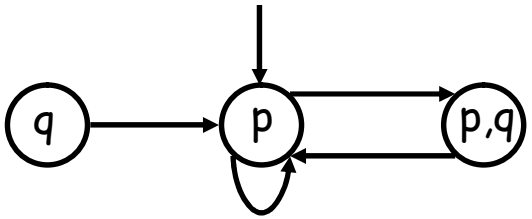


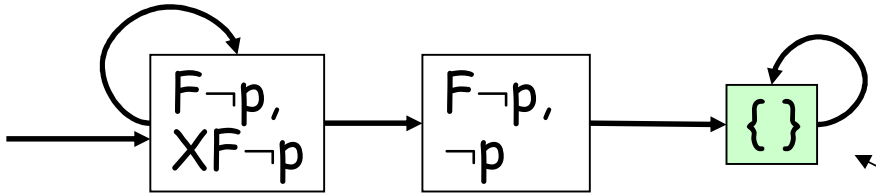


$\models E_F$ true
also gilt φ nicht in K

Beispiel

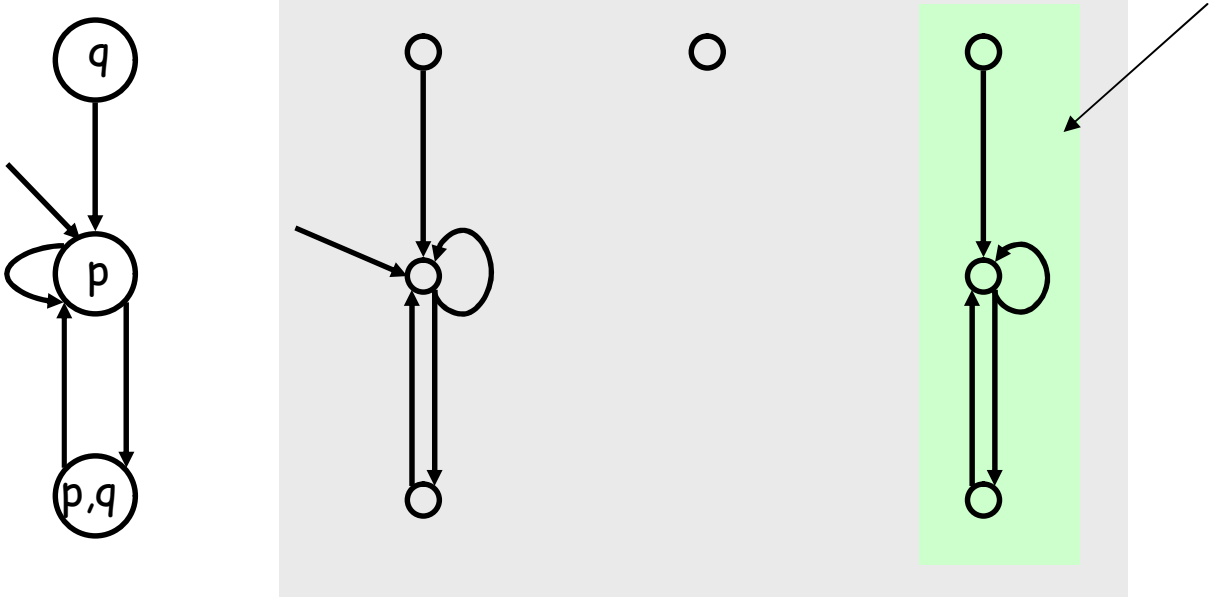
- Prüfe LTL-Formel $\varphi = G\ p$ in der Kripke-Struktur
- Negation $\neg\varphi = \neg\ G\ p$
- positive Normalform $\neg\varphi = F\neg\varphi$
- Tableau:





Produkt-
konstruktion

Fairnessmenge F

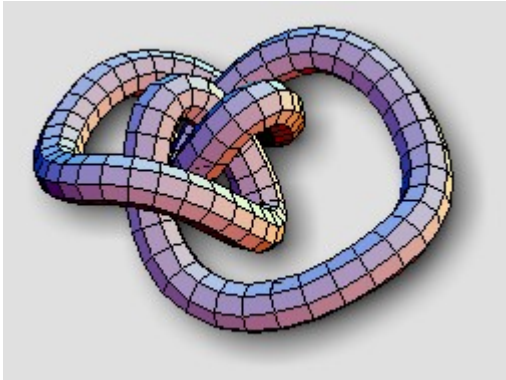


$\models A_F$ false
also gilt φ in K

Implizite Konstruktion

- Gegeben
  - Kripke Struktur K als SMV-Spezifikation P
  - Formel $A\varphi$
  - „Addiere" SMV-Spezifikation für Tableau $T_{\neg\varphi}$
  - Prüfe
    - EG true
    - FAIRNESS $\neg(\alpha_i U\ \beta_i) \vee \beta_i$
- Genaue Konstruktion siehe Clarke, Grumberg, Hamaguchi: *Another Look at LTL-Model Checking*

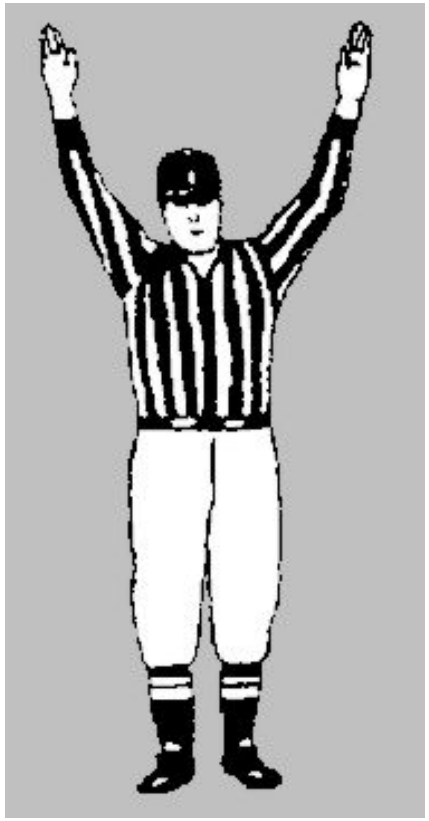
Komplexität



- $m=|S|$, $n = |\varphi|$
- LTL Model-Checking
  - $O(m)O(2^n)$
- CTL-Model-Checking
  - $O(m)O(n)$
- In der Praxis
  - m sehr groß
  - n klein

LTL oder CTL ?

- CTL
  - + einfacher zu implementieren
  - + Bewährte Systeme - SMV
  - + geringere Komplexität in φ
  - - Formeln komplizierter
- LTL
  - - Implementierung komplizierter
  - - grössere Komplexität in φ
  - + **aber**
    Formelgrösse nicht so entscheidend
  - + Es gibt effiziente Implementierungen
  - + gut verständlich
  - + Fairness ausdrückbar
  - + LTL in vielen Systemen implementiert
    SPIN, NuSMV, cadenceSMV
- Kompromiss:
  - Beides - siehe NuSMV
  - Erweiterungen: Sugar, Bandera, ...
- Zur Diskussion siehe: M. Vardi:
  - The Quest for the ultimate Temporal Specification Languages



CTL*-Model Checking

- Zurückführbar auf LTL-Model Checking
- Führe sukzessiv von innen nach außen Abkürzung für Teilausdrücke der Form $A\varphi$ bzw. $E\psi$ ein, wenn φ und ψ LTL-Formeln sind:
  - $x_i := A\varphi$ bzw. $x_i := A\,\neg\psi$
  - die Mengen $[x_i] = [A\varphi]$ bzw. $[\neg x_i] = \neg[A\neg\psi] = [E\psi]$ kann man durch LTL-Modelchecking bestimmen
- Beispiel: $\Psi = E\,GFp \vee EF\,AG\,q$
  1. Setze $x_0 := A\,\neg GFp$ und $x_1 := A\,Gq$, dann gilt: $\Psi = \neg x_0 \vee EF\,x_1$
  2. setze $x_2 := A\,\neg Fx_1$
  3. mit diesen Substitutionen gilt: $\Psi = \neg x_0 \vee \neg x_2$

$$\Psi = EGFp \vee EFAGq$$

$$= \text{let } [x_0 = A\neg GFp] \text{ in } \neg x_0 \vee EFAG\,q$$

$$= \text{let } [x_0 = A\neg GF\,p,\ x_1 = AGq] \text{ in } \neg x_0 \vee EFx_1$$

$$= \text{let } [x_0 = A\neg GFp,\ x_1 = AGq,\ x_2 = A\neg Fx_1] \text{ in } \neg x_0 \vee \neg x_2$$

$$= \text{let } [x_0 = AFG\neg p,\ x_1 = AGq,\ x_2 = AG\neg x_1] \text{ in } \neg x_0 \vee \neg x_2$$

CTL*-Model Checking allgemein

- Gegeben eine CTL*-Formel Ψ
- Substituiere sukzessive Teilausdrücke der Form
  - $A\varphi$ bzw. $E\psi$ wobei ϕ bzw ψ reine LTL-Formeln sind durch neue Variablen $x_i := A\phi$ bzw. $x_j := A\neg\psi$
  - Aus Ψ entsteht auf diese Weise ein rein aussagenlogischer Ausdruck α mit Variablen $\{x_1,\ldots,x_n\}$, deren Gültigkeitsbereiche durch LTL-Formeln beschrieben werden

$$\Psi = \text{let } [\, x_1 := A\phi_1, \ldots, x_n := A\phi_n \,] \text{ in } \alpha$$

wobei die ϕ_i LTL-Formeln in den Variablen

$$AP \cup \{x_1,\ldots,x_n\}$$

sind.

- Damit ist CTL*-Model Checking i.W. auf LTL-Model-Checking zurückgeführt