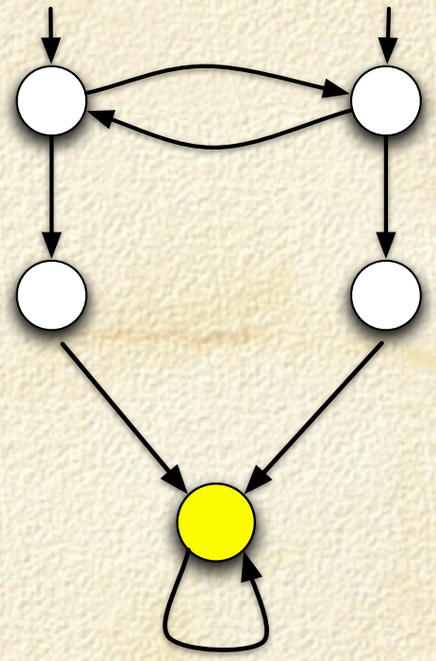
Definition 1.23 *(Kripke-Struktur 2)*



Eine Kripke-Struktur $M := (S, S_0, R, E_S)$ *besteht aus*

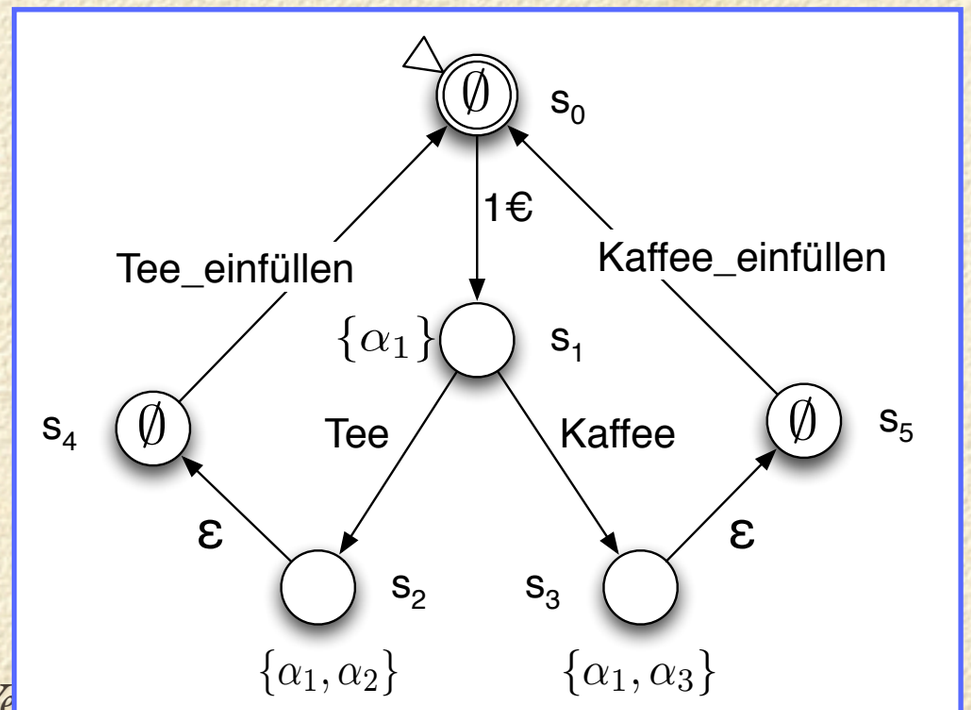
a) einer endlichen Zustandsmenge S,

b) einer Menge $S_0 \subseteq S$ *von Anfangszuständen,*

c) einer linkstotalen *(Transitions-)Relation* $R \subseteq S \times S$ *und*

d) einer Zustandsetikettenfunktion $E_S : S \to \mathcal{P}(AP)$*, die jedem Zustand* s *eine Menge* $E_S(s) \subseteq AP$ *von aussagenlogischen atomaren Formeln zuordnet (die in diesem Zustand gelten).*

$$\forall s \in S \; \exists s' \in S : (s, s') \in R$$

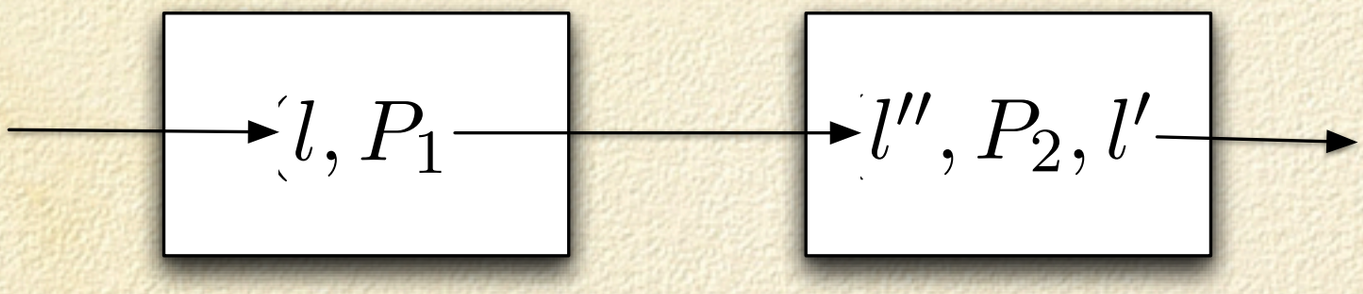


Zuweisung:

$$\mathcal{C}(l, v \leftarrow e, l') \equiv \quad pc = l \wedge pc' = l' \wedge v' = e \wedge same(V \backslash \{v\})$$

Hintereinanderausführung:

$$\mathcal{C}(l, \boxed{(P_1; l'' : P_2)}\ , l') \equiv \mathcal{C}(l, \boxed{P_1}, l'') \vee \mathcal{C}(l'', \boxed{P_2}, l')$$



Anfangswert $x = 1, y = 2$.

$l_1 : x := 2 \cdot y;$

$l_2 : y := y - 1;$

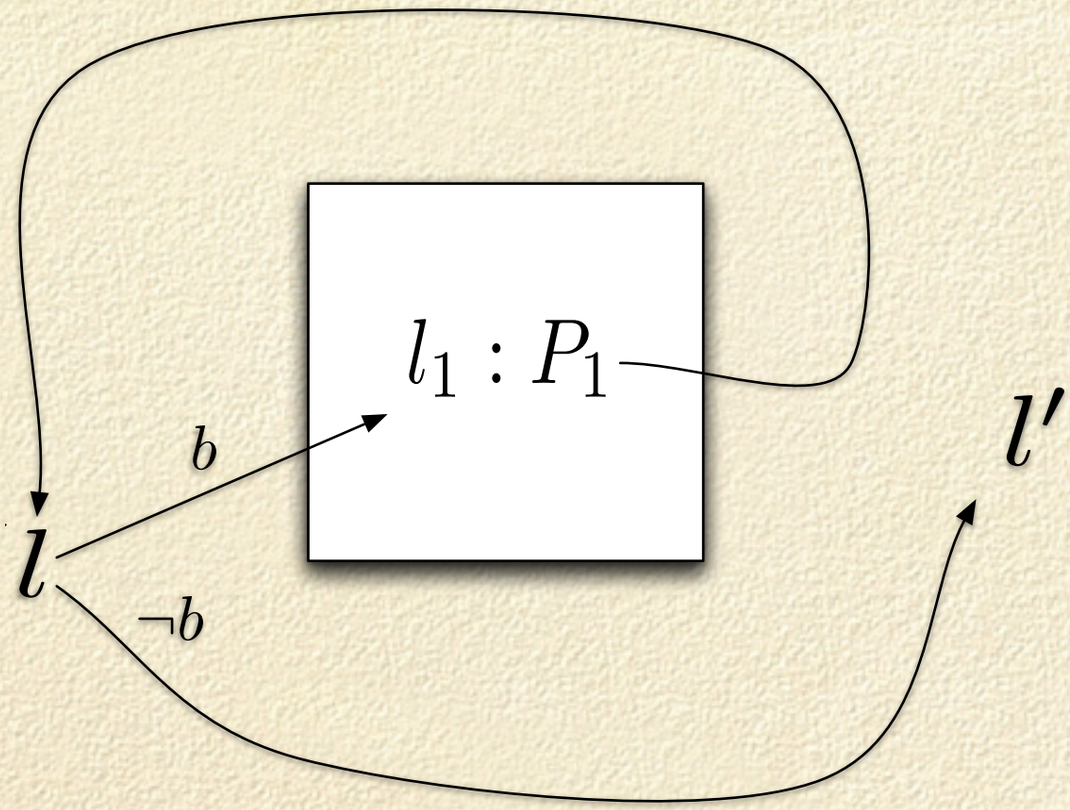
l_3

$$\mathcal{S}_0 \equiv (x = 1 \wedge y = 2 \wedge pc = l_1)$$

Beispiel 1.28 auf Seite 28

Schleifen-Anweisung:

$$\mathcal{C}(l,\ \textbf{while}\ b\ \textbf{do}\ l_1 : P_1\ \textbf{endwhile},\ l') \equiv$$
$$(pc = l \wedge pc' = l_1 \wedge b \wedge same(V)) \vee$$
$$(pc = l \wedge pc' = l' \wedge \neg b \wedge same(V)) \vee$$
$$\mathcal{C}(l_1, P_1, l)$$

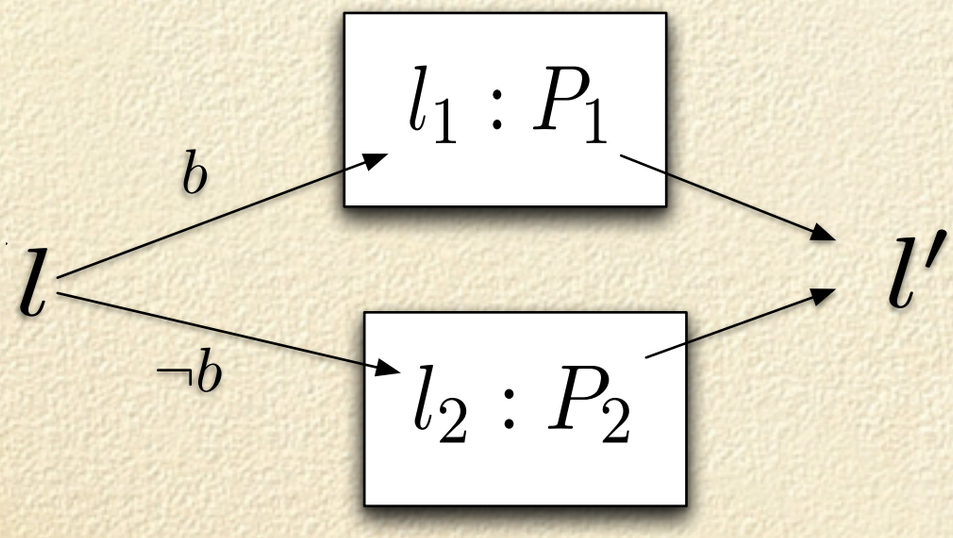


Skip:

$$\mathcal{C}(l, skip, l') \equiv pc = l \wedge pc' = l' \wedge same(V)$$

Bedingte Anweisung:

$$\mathcal{C}(l, \textbf{ if } b \textbf{ then } l_1 : P_1 \textbf{ else } l_2 : P_2 \textbf{ endif}, l') \equiv$$
$$(pc = l \wedge pc' = l_1 \wedge b \wedge same(V)) \vee$$
$$(pc = l \wedge pc' = l_2 \wedge \neg b \wedge same(V)) \vee$$
$$\mathcal{C}(l_1, P_1, l') \vee \mathcal{C}(l_2, P_2, l')$$



$$P^{\mathcal{L}} = \textbf{cobegin}\ l_1 : P_1^{\mathcal{L}}\ l_1' \| \ldots \| l_n : P_n^{\mathcal{L}}\ l_n'; \textbf{coend}$$

$$\mathcal{C}(l, P^{\mathcal{L}}, l') \equiv$$
$$(pc = l \wedge pc_1' = l_1 \wedge \ldots \wedge pc_n' = l_n \wedge pc' = \bot) \vee$$

Initialisierung

$$(pc = \bot \wedge pc_1 = l_1' \wedge \ldots \wedge pc_n = l_n' \wedge pc' = l' \wedge \textstyle\bigwedge_{i=1}^{n}(pc_i' = \bot)) \vee$$

Termination

$$(\textstyle\bigvee_{i=1}^{n}(\mathcal{C}(l_i, P_i, l_i') \wedge same(V \backslash V_i) \wedge same(PC \backslash \{pc_i\})))$$

Transition von P_i

await b

warte bis b *gilt !*

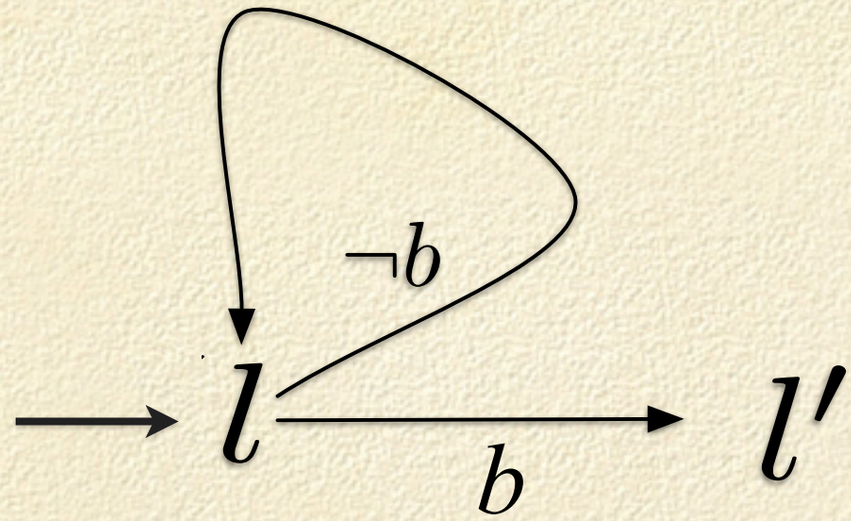
await-Anweisung:

$$\mathcal{C}(l, \mathbf{await}(b), l') \equiv$$
$$(pc_i = l \wedge pc_i' = l \wedge \neg b \wedge same(V_i))$$
$$\vee\ (pc_i = l \wedge pc_i' = l' \wedge b \wedge same(V_i))$$

"busy waiting"

b = false

b = true



Beispielprogramm für wechselseitigen Ausschluss

$$P = m : \mathbf{cobegin}\ P_0 \| \ P_1\ \mathbf{coend}$$

Parallele Programme oder Prozesse können zum konsistenten Schreiben auf gemeinsame Daten einen „kritischen Abschnitt“ enthalten, der nicht überlappend ausgeführt werden darf. Dies kommt im „nicht-kritischen Abschnitt“ nicht vor.

Die Programme sollen dabei (für den Fall zweier Prozesse P und Q) folgende Eigenschaften erfüllen:

A) Die Befehlszähler von P und Q sind nie gleichzeitig in ihren kritischen Abschnitten.

Markierungs-Invarianz

safety property

Es gilt immer ....

B) Meldet der Prozess P oder Q den Wunsch zum Eintritt in den kritischen Abschnitt an ($wantP = True$ oder $wantQ = True$), so kann er nach einer gewissen endlichen Zeit tatsächlich in seinen kritischen Abschnitt eintreten.

Lebendigkeits-Invarianz

liveness property

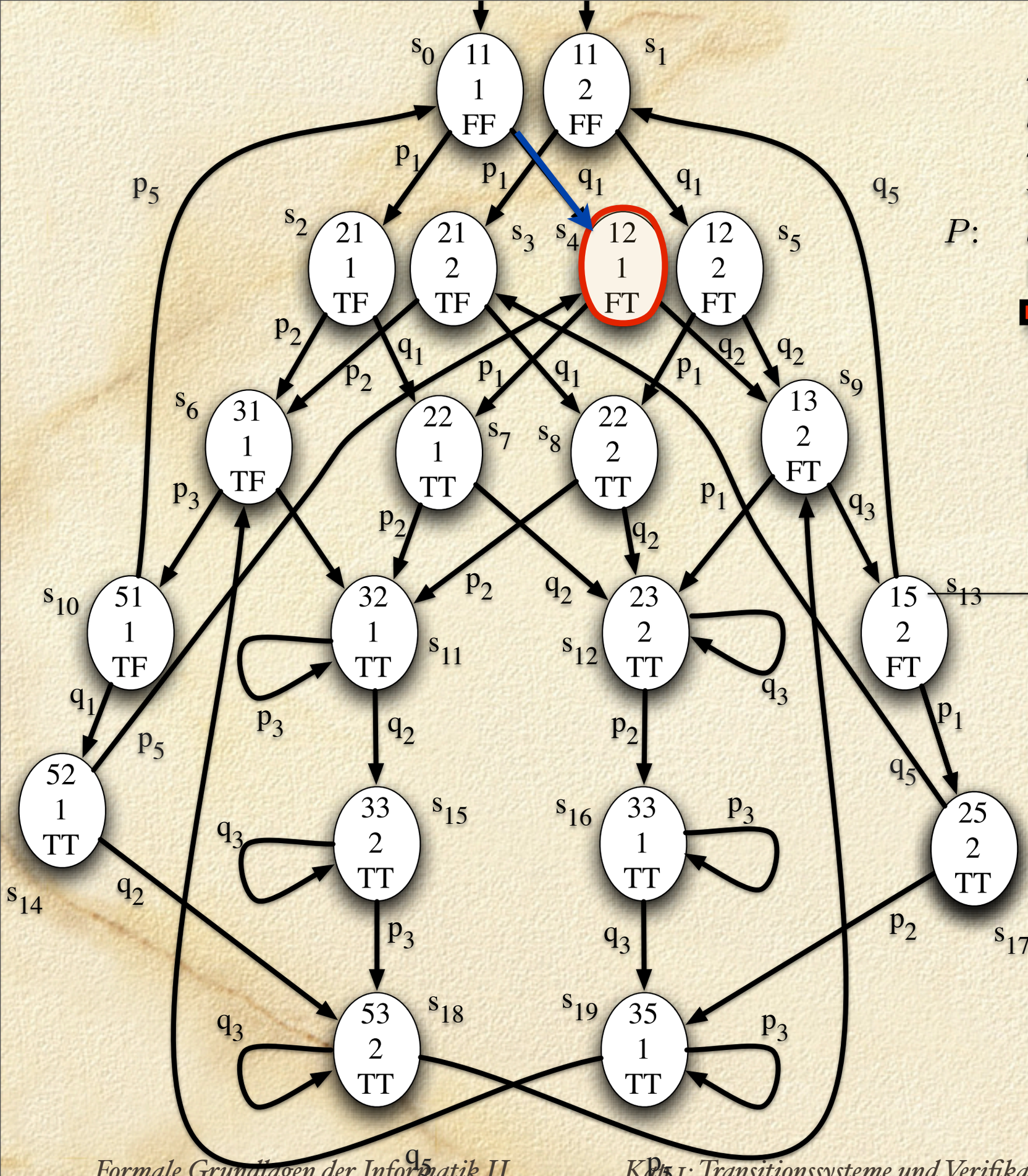
Es gilt später einmal ....

allgemeines Programm-Schema

Initialisierung
m : **cobegin** $P \| Q$ **coend**
wobei

P: l_0 : **while** True **do**
p_i : non-critical section;
$Eintrittsprotokoll$
p_j : critical section;
$Austrittsprotokoll$
endwhile l'_0

Q: l_1 : **while** True **do**
q_i : non-critical section;
$Eintrittsprotokoll$
q_j : critical section;
: $Austrittsprotokoll$
endwhile l'_1



$wantP = wantQ = False : boolean,$
$last = 1 \vee last = 2 : integer,$
m : **cobegin** $P \| Q$ **coend**
wobei

P: l_0 : **while** True **do**

[p_0 : non critical section;]

$p_1 : wantP := True;$
$p_2 : last := 1;$
p_3 :**await**$(wantQ = False \vee last = 2);$

[p_4 : critical section;]

$p_5 : wantP := False;$
endwhile l_0'