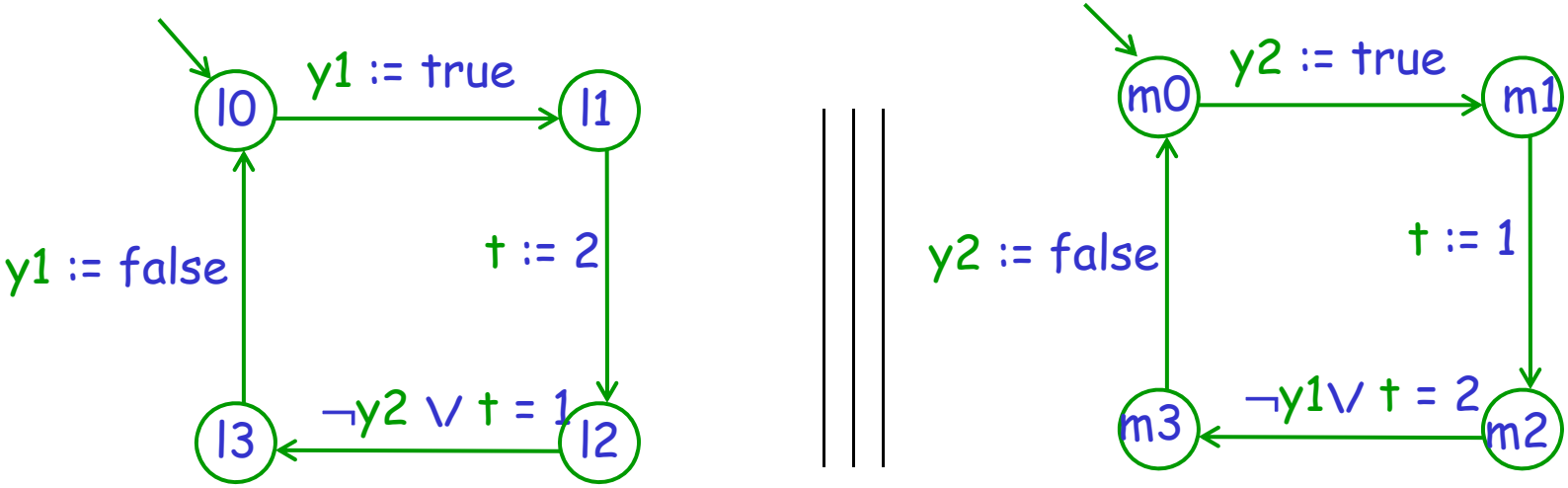
Lecture 4

Properties of Transition Systems

Peterson-Fischer: Possible Specifications

Variables: y1, y2: boolean, t: {1,2}
Initially y1 = y2 = false, t = 1



Mutual Exclusion: the two processes never simultaneously reach l3,m3

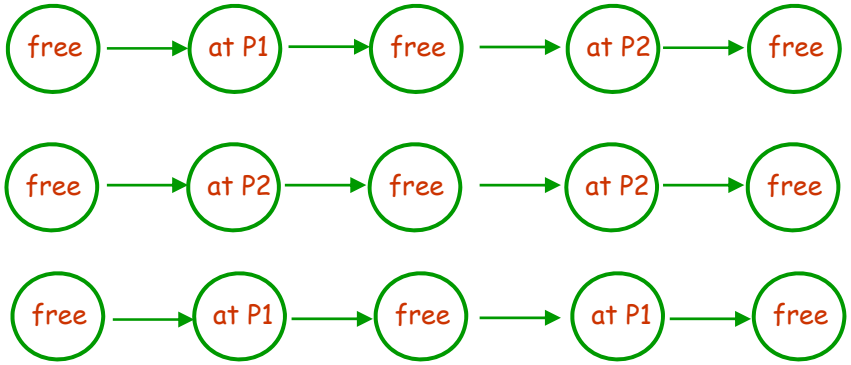
Absence of Starvation: If the left process is at l1, it will later reach l3

Bounded Overtaking: If the left process is at l1, the other process will reach m3 at most once (twice?) before the left process reaches l3

Two Classes of Formalizations

Linear Time Properties:

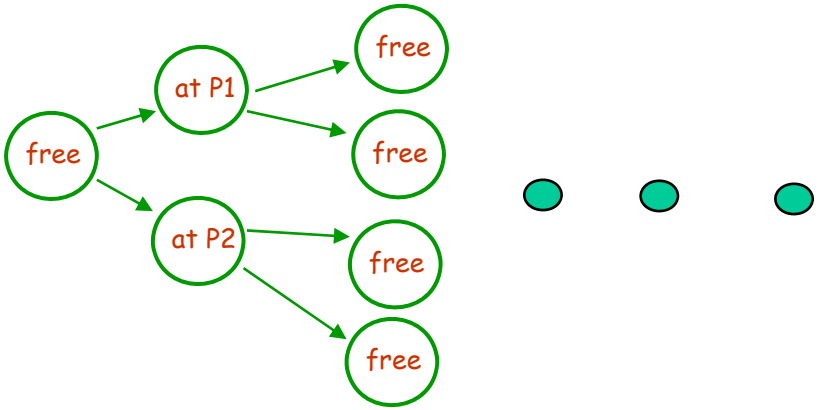
Specify properties that hold for all computations



Free in every 2nd state ;
at P1 always followed by free

Branching Time Properties:

Specify properties that hold for the computation tree (unfolding)



always possible to reach free ;
always possible to reach at P1 ;
free is always eventually reached

Transition Systems (formal definition)

A Transition System is a tuple $(S, S_0, \rightarrow, V, L)$ where

- S is a set of states
- S_0 is a set of initial states
- $\rightarrow$ (a subset of $S \times S$) is a transition relation
- V is a set of variables
- L : $S \rightarrow (V \rightarrow Val)$ gives a valuation of variables in each state

Write $s \rightarrow s'$ for $(s, s') \in \rightarrow$

A Computation is a finite or infinite sequence of states

s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9

such that

- s_o is an initial state
- $(s_i, s_{i+1}) \in \rightarrow$ for all (relevant) i

Linear Time Properties

- Properties of Computations
- Specify ”what happens when the system executes”
- Technical convenience
  - Consider only infinite computations
  - For finite computations, repeat the last state:

    s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7

    becomes

    s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_7 s_7 s_7 .....
- In essence

  Linear time property φ = set of computations
- Transtion system T satisfies linear time property φ

  $T \models \varphi$

  if all computations of T are in (satisfy) φ

Linear Time Properties: examples

- Start from properties about states p , q

  e.g., y0 < 1 x = y at m3

- p is invariant

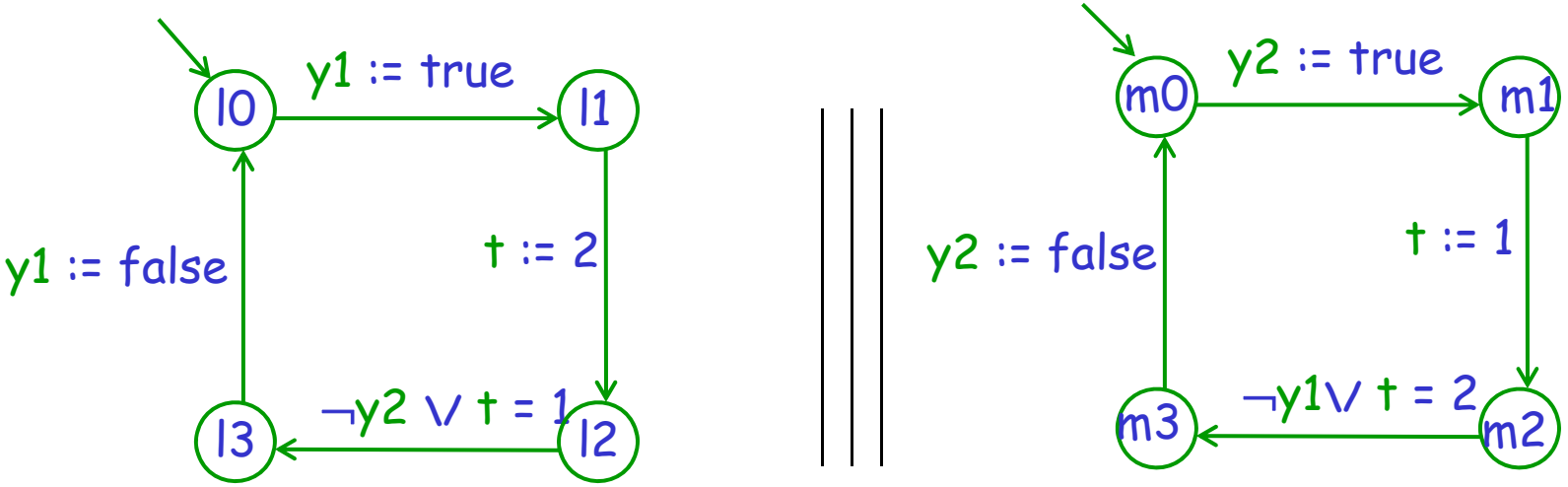
- p is stable

- p precedes q

- p leads to q

Peterson-Fischer: Possible Specifications

Variables: y1, y2: boolean, t: {1,2}
Initially y1 = y2 = false, t = 1



Mutual Exclusion: ¬(at l3 /\ at m3) is invariant

Absence of Starvation: at l1 leads to at l3

Bounded Overtaking: ?????

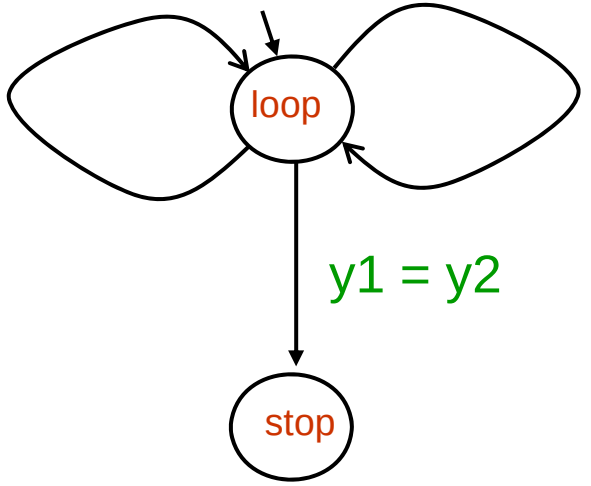
Example: GCD Computation

Action System
Variables: y1,y2: integer
Initially y1=x1, y2=x2

P:

y1 > y2 ->
y1 := y1-y2

y1 < y2 ->
y2 := y2-y1



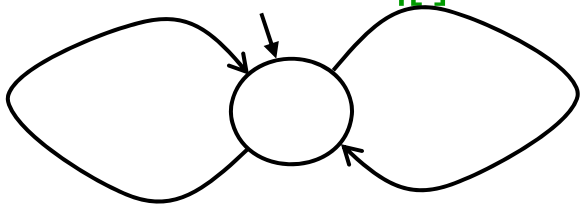
Partial Correctness:
x1> 0 /\ x2> 0 ->
[P@stop -> y1=y2=gcd(x1,x2)] is invariant

Total Correctness:
x1> 0 /\ x2> 0 ->
[P@loop leads to P@stop]

Example: Termination Detection

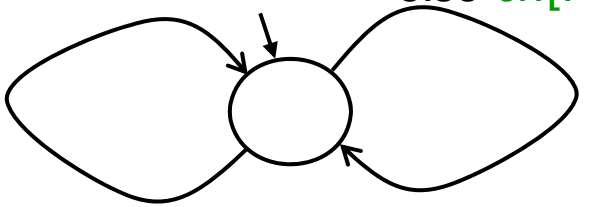
Variables: ch[N]: FIFO Channel of {black,white}
q[N]: boolean
dist = false: boolean
Processes: Q[0] || ... || Q[N-1] || P[0] || ... || P[N-1]

Q[i]:



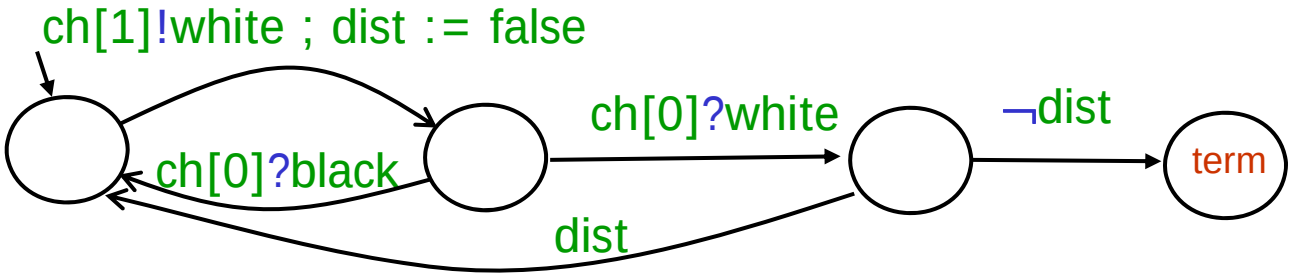
q[i-1] -> q[i] := false ;
if i=0 then dist := true

P[i]:



ch[i]?black -> ch[i+1]!black

P[0]:



Example: Termination Detection

Abbreviation:

$$Terminated \equiv \forall i : 0 \leq i < N :: q[i]$$

Safety Property:

$$[P[0]@term \rightarrow Terminated\ ] \quad \text{is invariant}$$

Liveness Property:

$$Terminated \quad \text{leads to} \quad P[0]@term$$