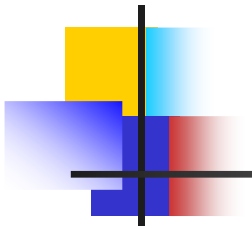
Model checking with Message Sequence Charts

Doron Peled

Collaborators: R. Alur, E. Gunter, G. Holzmann, A. Muscholl, Z. Su

Department of Computer Science
University of Warwick, UK

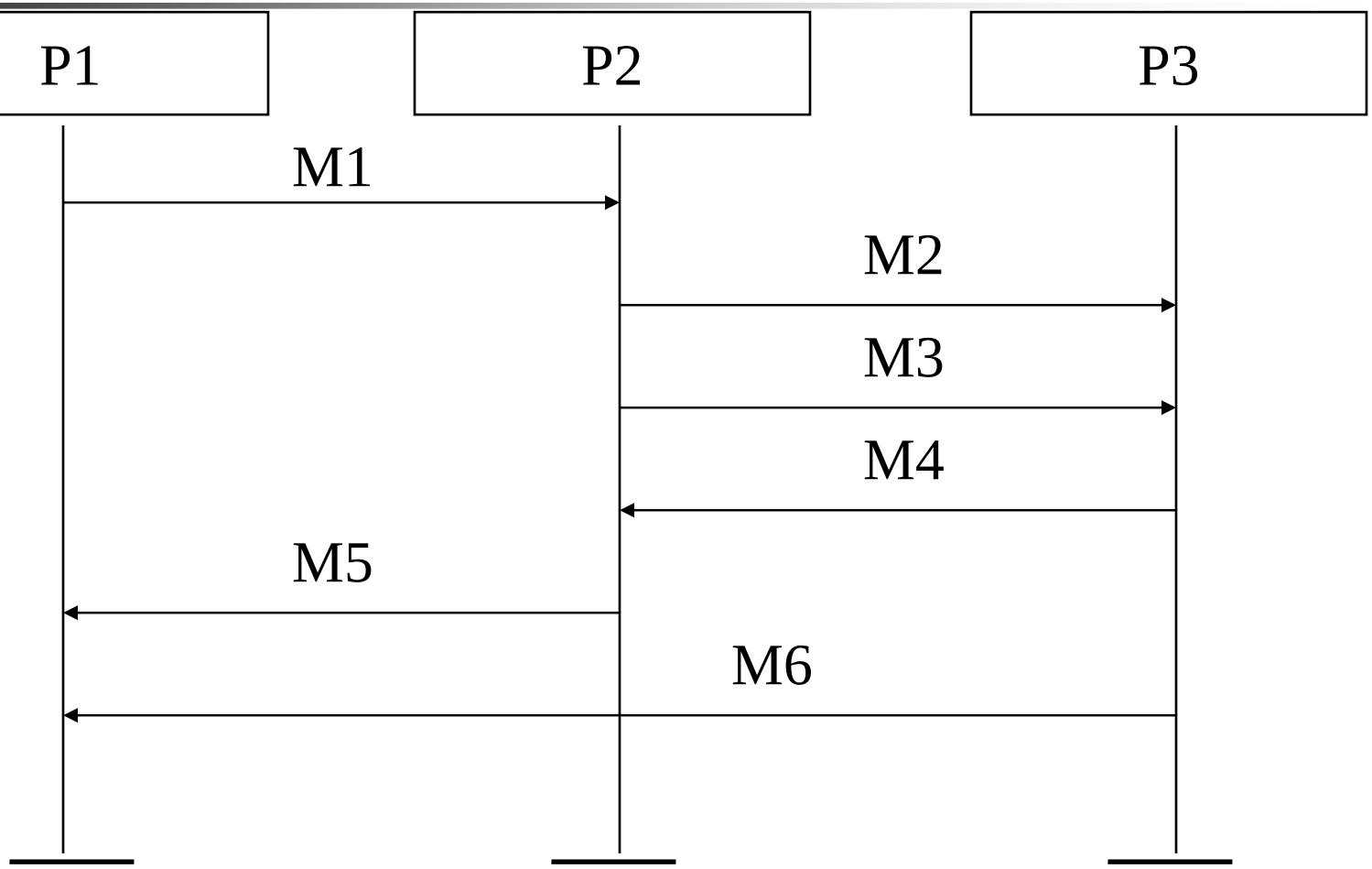
Bar Ilan University, IL



MSCs

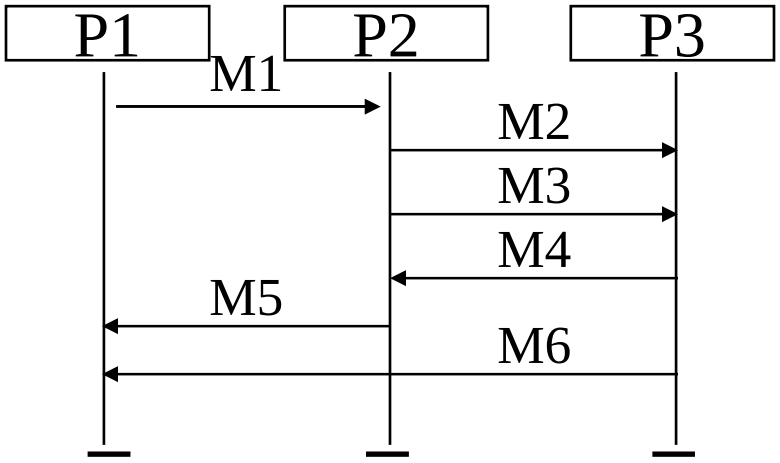
- An ITU standard notation (Z120).
- Visual + Textual forms.
- Specifies behaviors of communication protocols.
- Existing algorithms + tools.

MSC visual notation



MSC Textual form

```
msc MSC;
inst P1: process Root,
     P2: process Root,
     P3: process Root;
  instance P1;
     out M1 to P2;
     in M5 from P2;
     in M6 from P3;
  endinstance;
  instance P2;
     in M1 from P1;
     out M2 to P3;
     out M3 to P3;
     in M4 from P3;
     out M5 to P1;
  endinstance;
  instance P3;
     in M2 from P2;
     in M3 from P2;
     out M4 to P2;
     out M6 to P1;
  endinstance;
endmsc;
```



Partial order semantics: How exactly?

V - Set of events (vertices).

N - names.

P - processes.

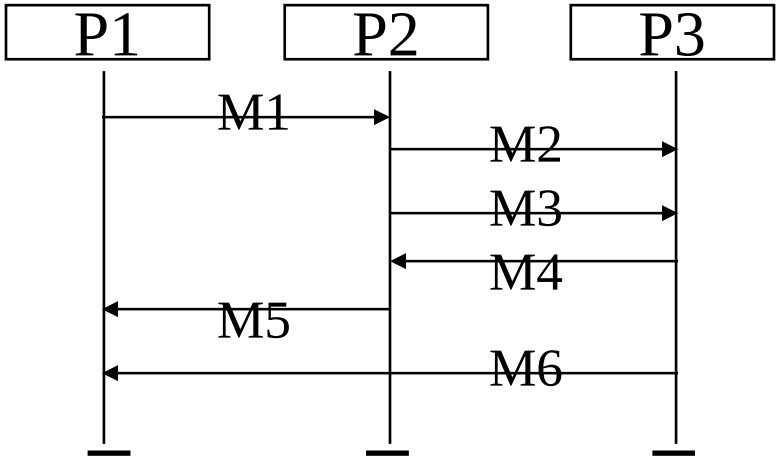
p:V→*P* maps events to processes.

E - Set of edges (orders, to be fixed).

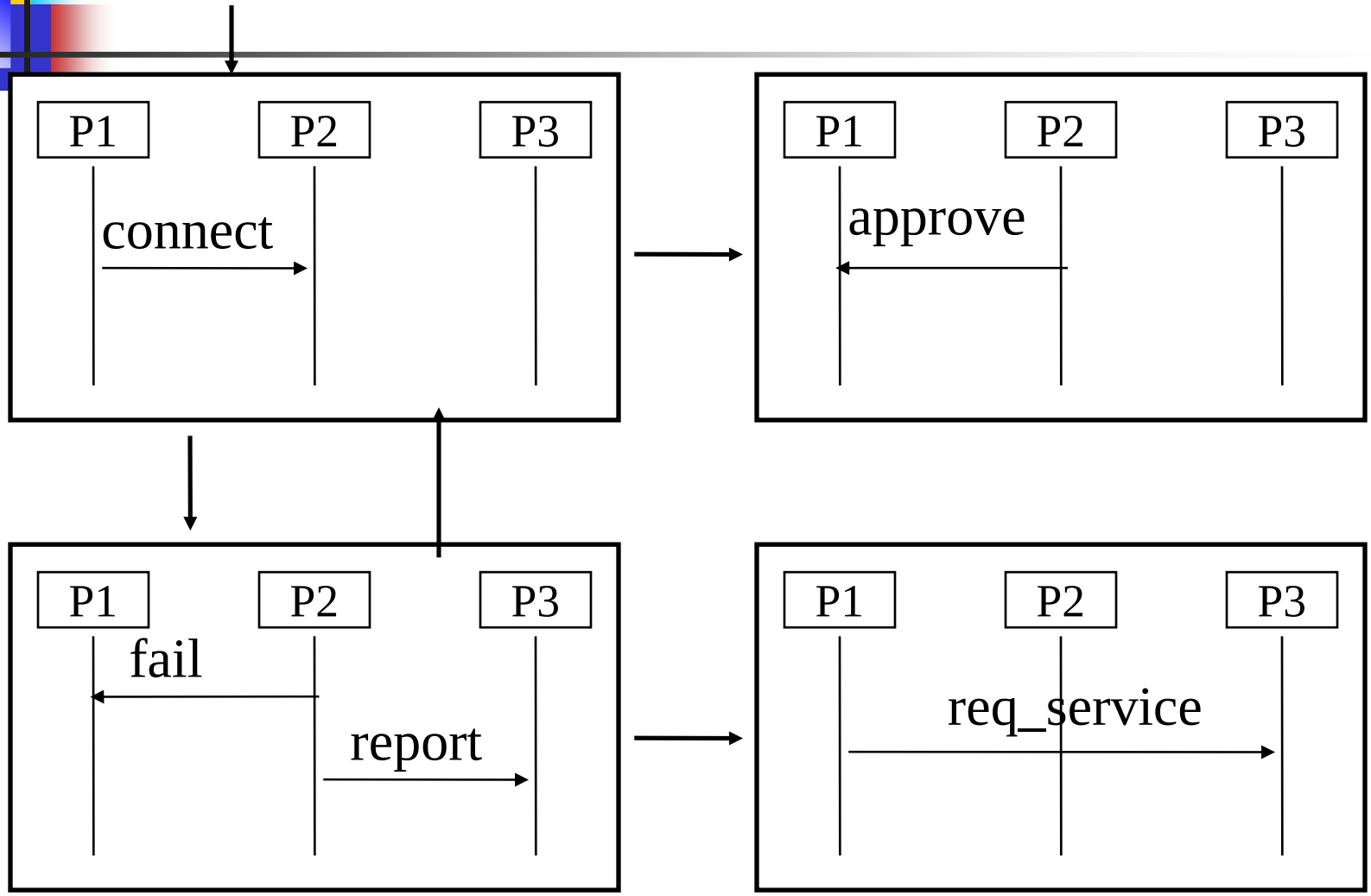
n: V→*N* naming function.

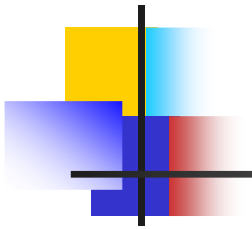
t: V → {s, r, l} matches a type (send, receive, local).

m⊆VxV - matching of sends and receives with same name and opposite types.



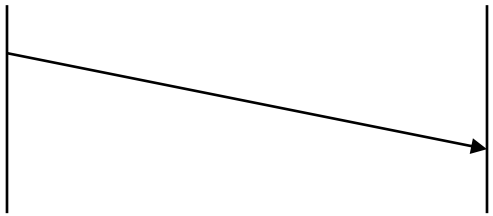
HMSCs: a graph of MSCs (again, what semantics?)



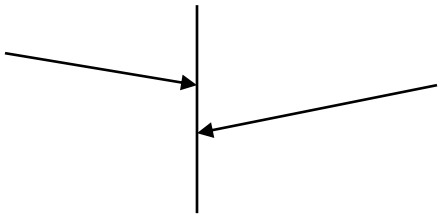


Visual semantics

- Sends before corresponding receives.

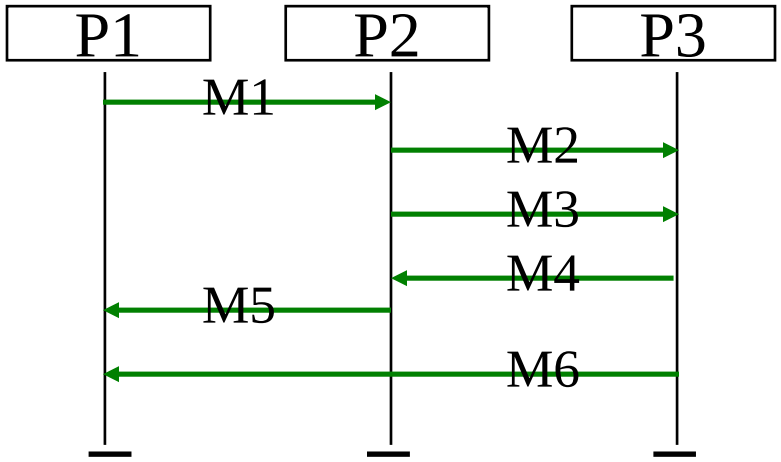


- Events on the same process line execute in order of appearance, from top to bottom.

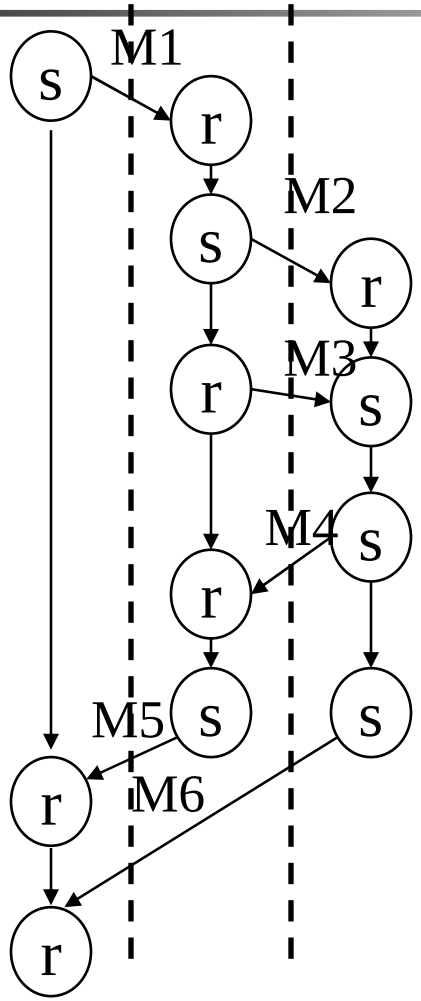


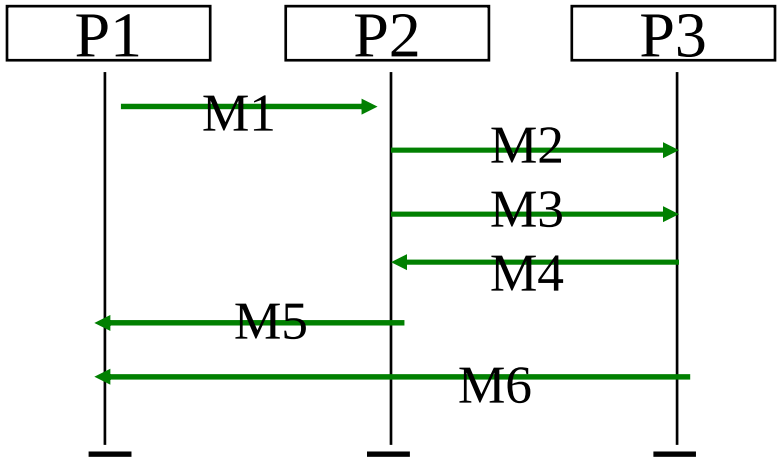
Visual order (wysiwyg)

- If some event (send, receive) is higher on the line than another, it comes first.
- Sends precede matching receives.



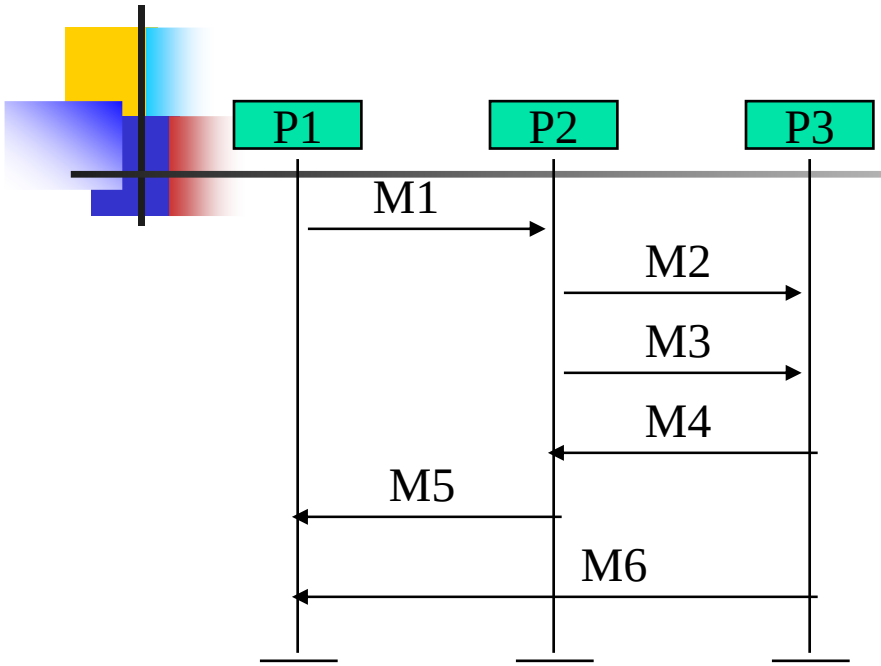
Visual order (wysiwyg)

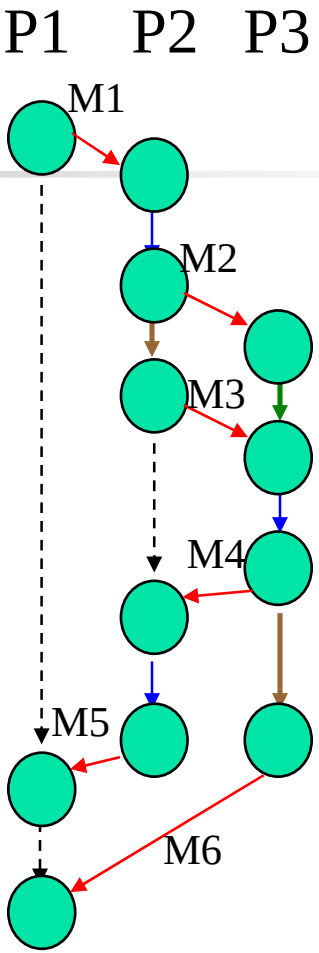




An events that appear in the process line above the other will precede it in time!

Causal Order

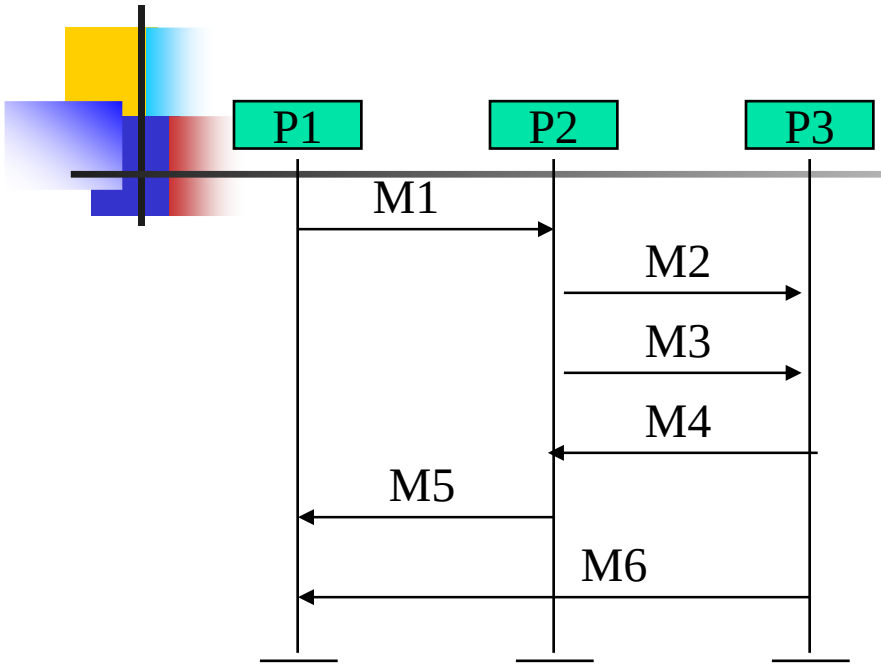


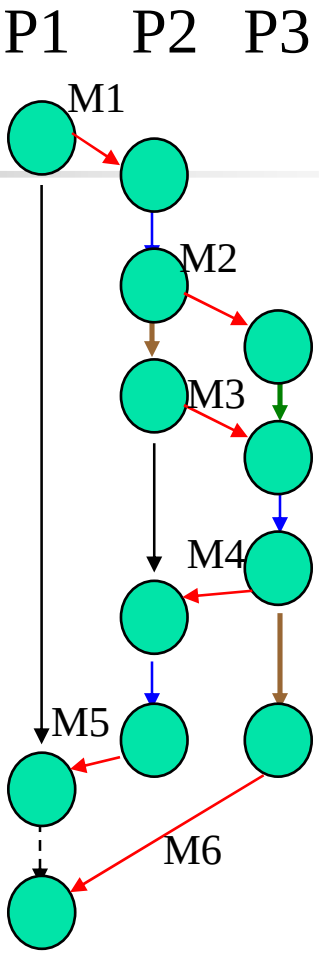


Rules: order between

- receive and a later send.
- two sends from same process.
- send and corresponding receive.
-fifo order.
-Everything that follows by transitive closure

Causal Order

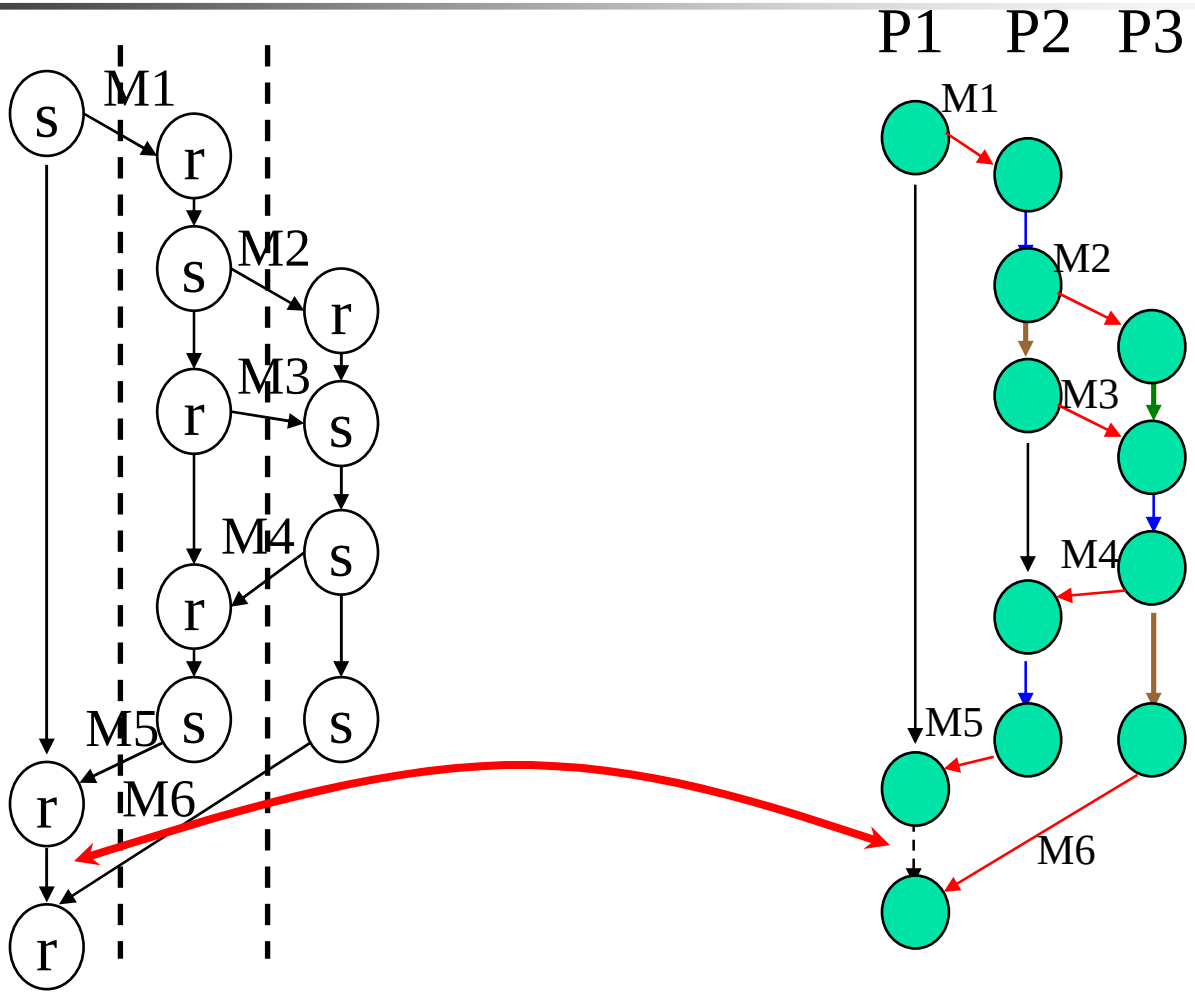




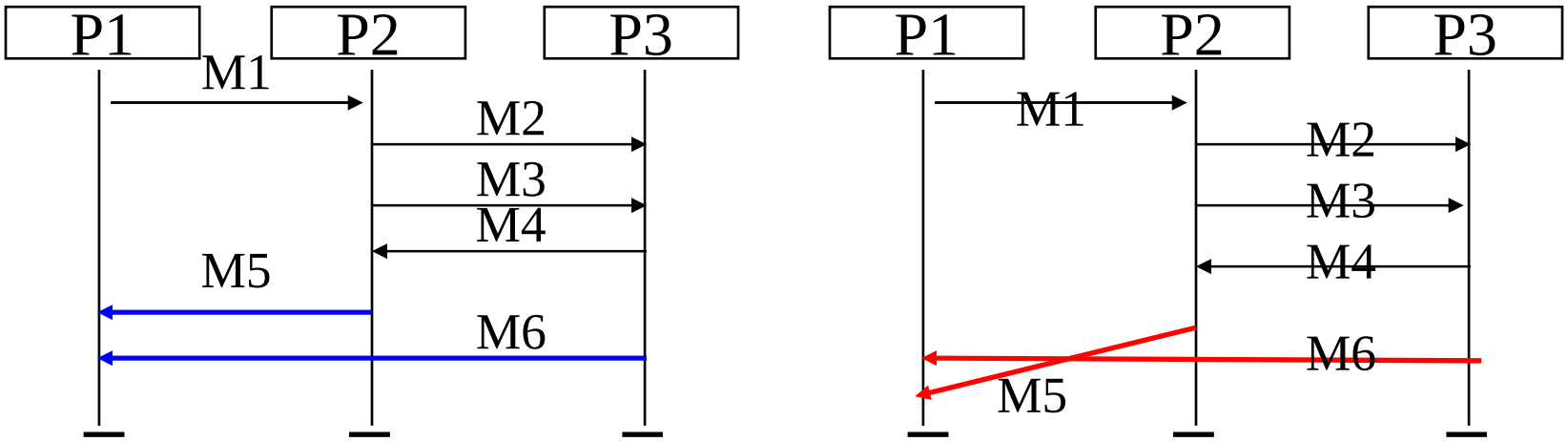
Rules: order between

- receive and a later send.
- two sends from same process.
- send and corresponding receive.
-fifo order.
-Everything that follows by transitive closure

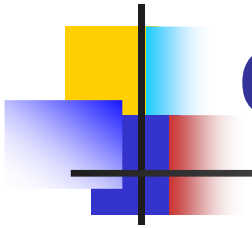
Races: the difference between the visual order and the causal order



Races: events can also occur in reverse order

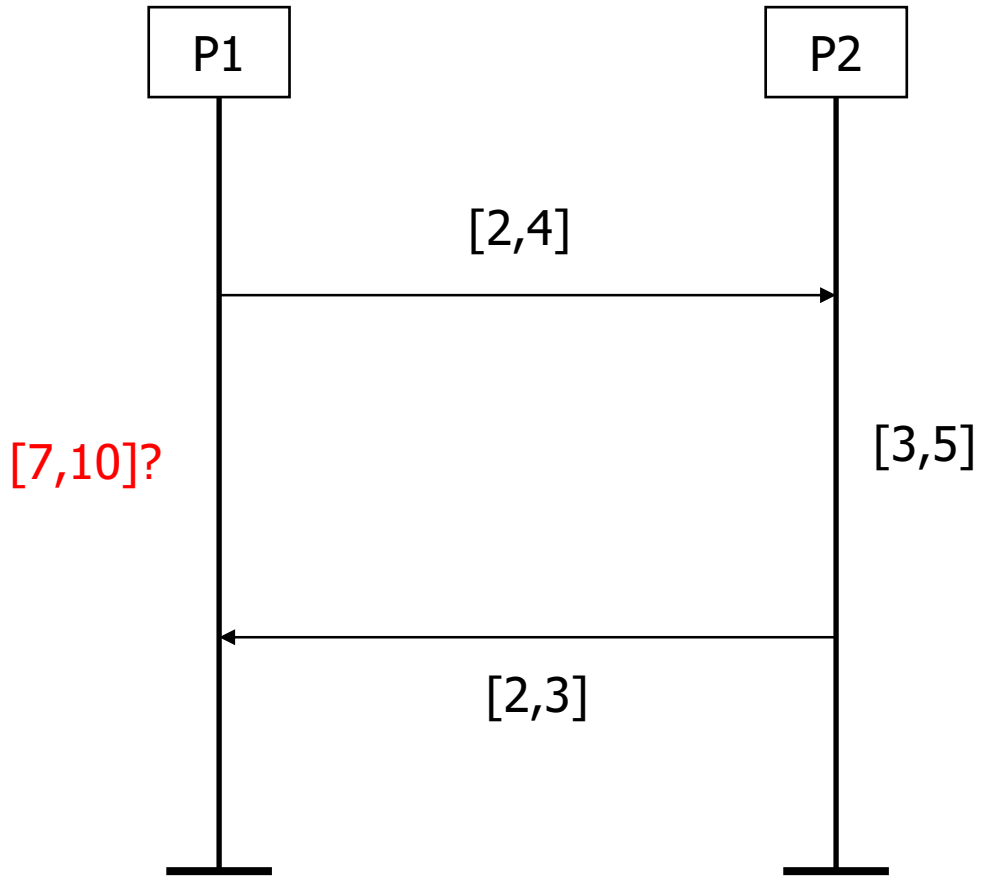


Calculating the transitive closure



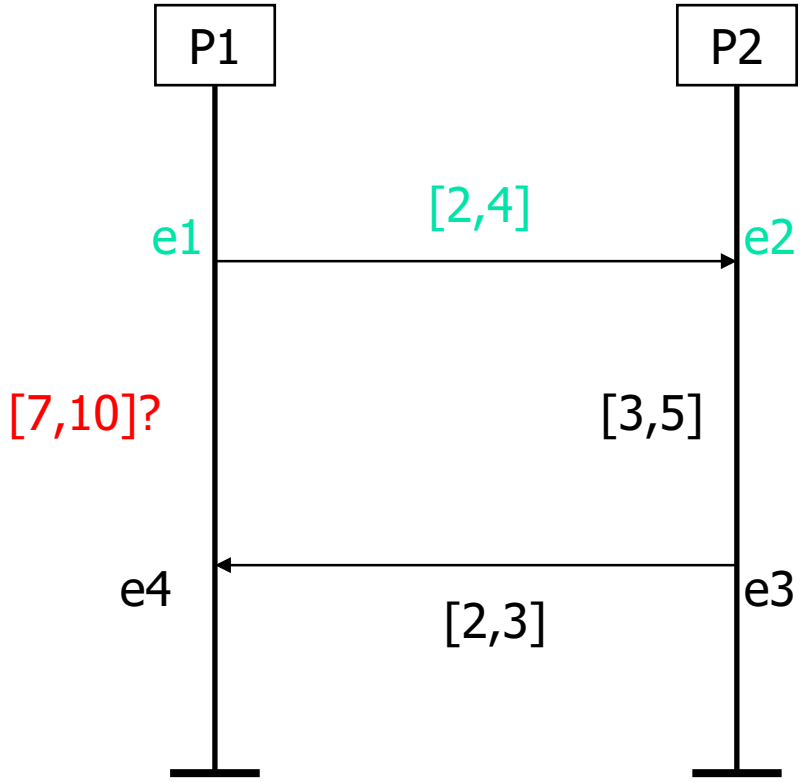
- Structure (E, R).
- E – Events, $R \subseteq E \times E$.
- R^* The transitive closure. Defined as follows:
  a R^*b if there is a sequence
  $x_1\ x_2\ \ldots\ x_n$ where $a=x_1$, $b=x_n$,
  and $x_i\ R\ x_{i+1}$ for $1 \le i < n$.
- Complexity: cubic. In our case: quadratic since every event has 1 or 2 successors).

Can also deal with time



Use Difference Bound Matrices (DBMs)

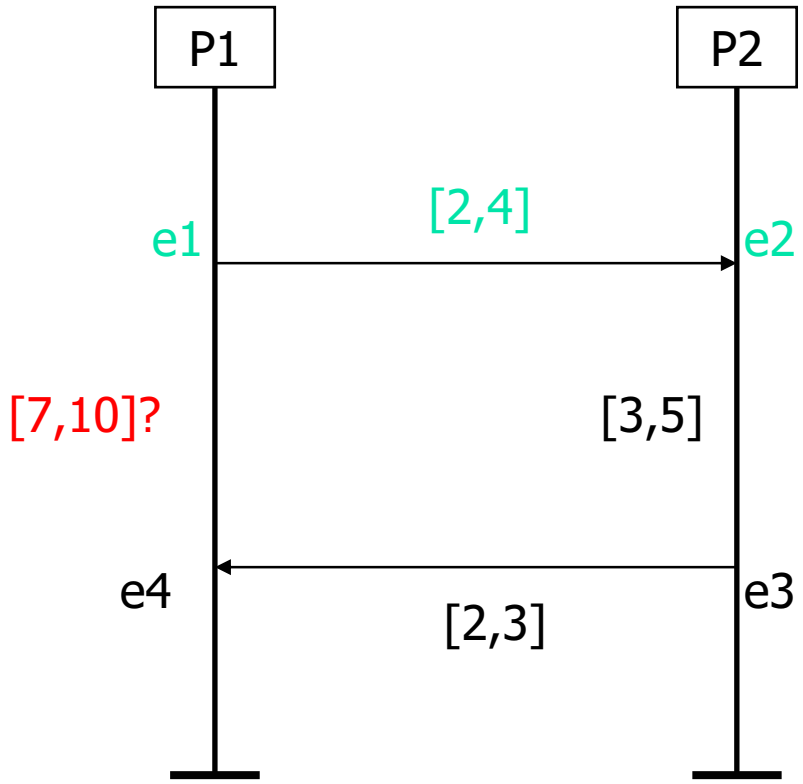
Put all known constraints, e.g., 2≤e2-e1≤4➔e2-e1≤4,e1-e2≤-2



	e1	e2	e3	e4
e1	0	4	◎	◎
e2	-2	0	5	◎
e3	◎	-3	0	3
e4	◎	◎	-2	0

Normalize: $e_i - e_j \le l + e_j - e_k \le m + e_i - e_k \le n$

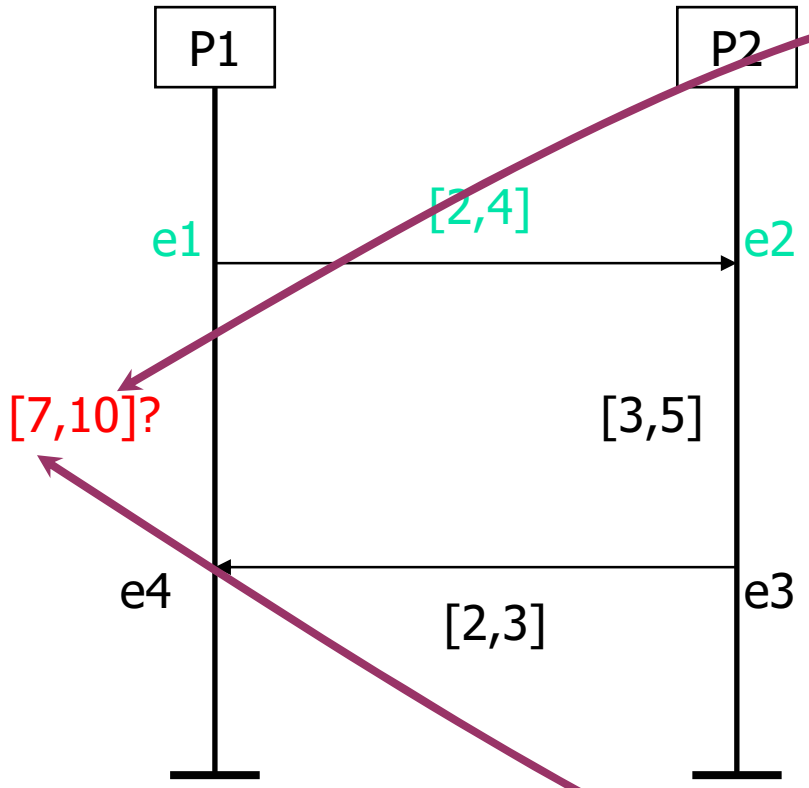
➔ $e_i - e_k \le min(n, l+k)$



9

	e1	e2	e3	e4
e1	0	4	◎	◎
e2	-2	0	5	◎
e3	◎	-3	0	3
e4	◎	◎	-2	0

Compare checked limit to normalized table.

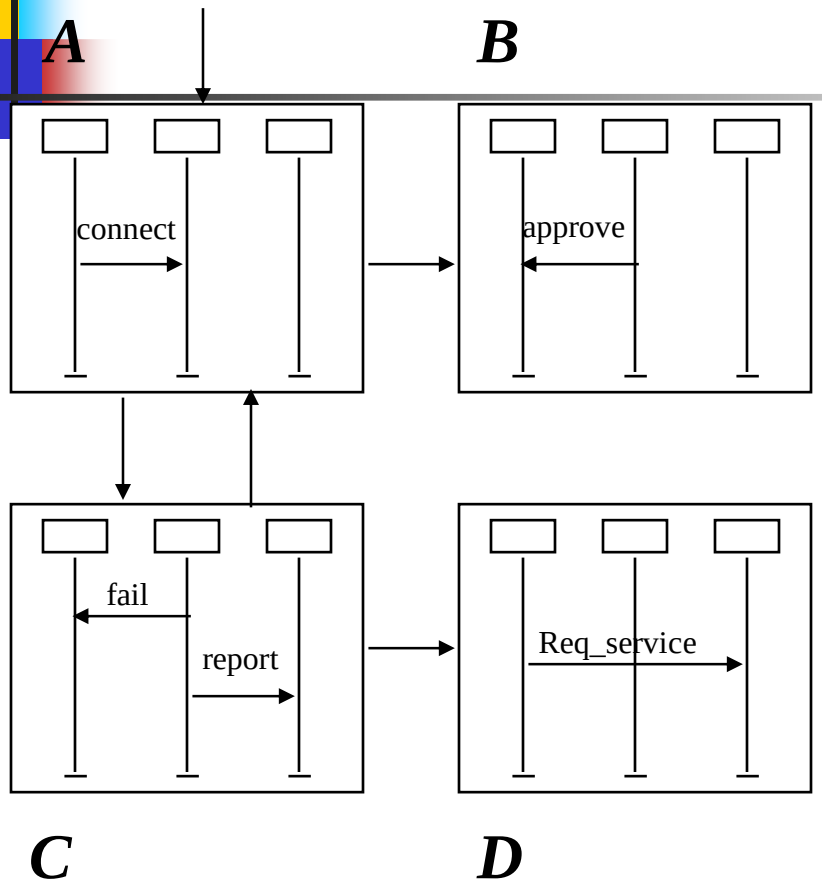


Problem!!

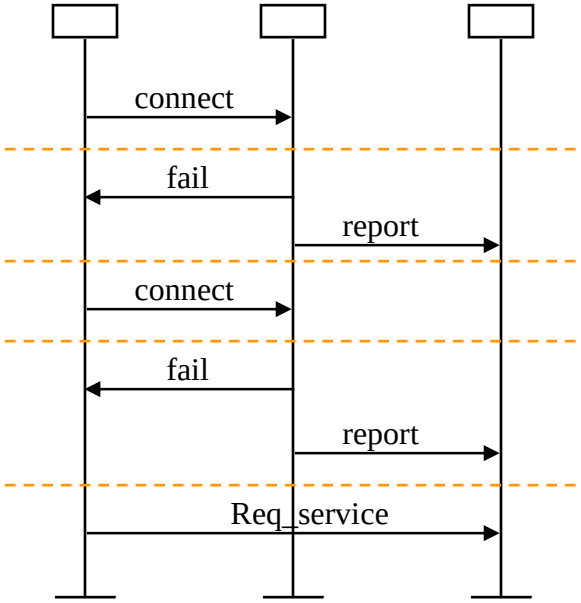
OK!!

	e1	e2	e3	e4
e1	0	4	9	12
e2	-2	0	5	8
e3	-5	-3	0	3
e4	-7	-5	-2	0

Visual concatenation semantics: extend process lines.
An execution: infinite or maximal



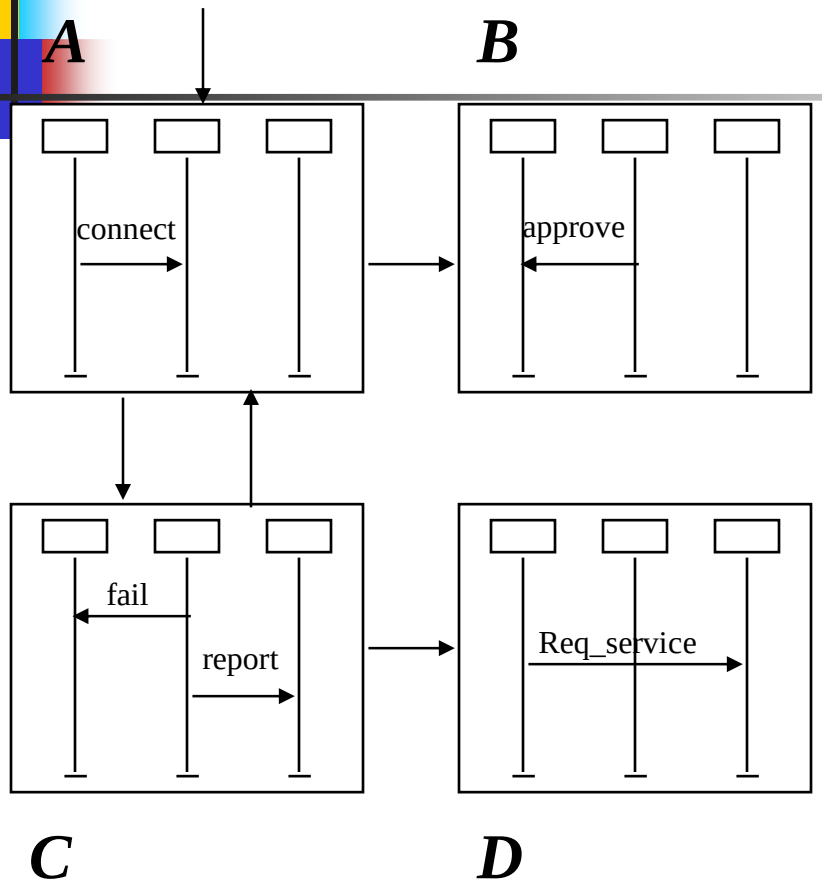
Execution: ***ACACD***



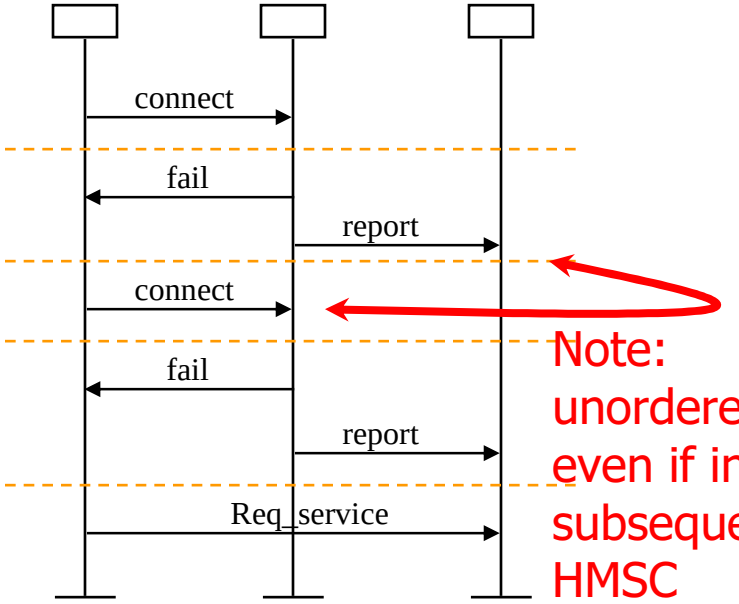
Technically: for concatenation: take union of events. For each process P_i, order events of A before events of C. For the path: take limit of union (A U AC U ACA …)

Visual concatenation semantics: extend process lines.

An execution: infinite or maximal



Execution: ***ACACD***

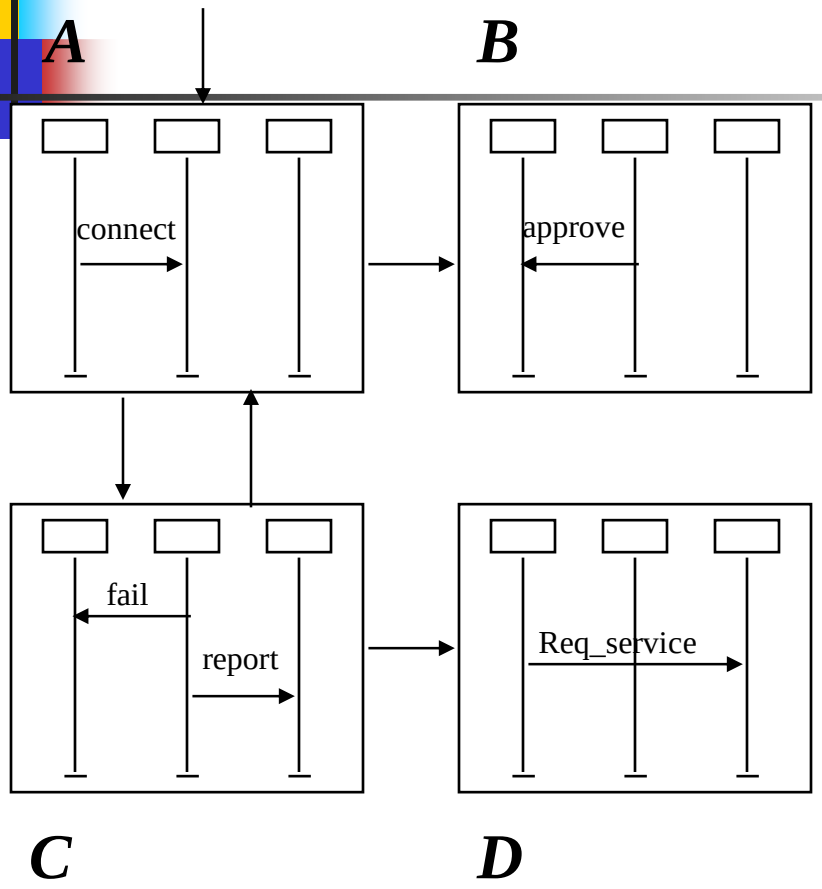


Note: unordered even if in subsequent HMSC boxes!!

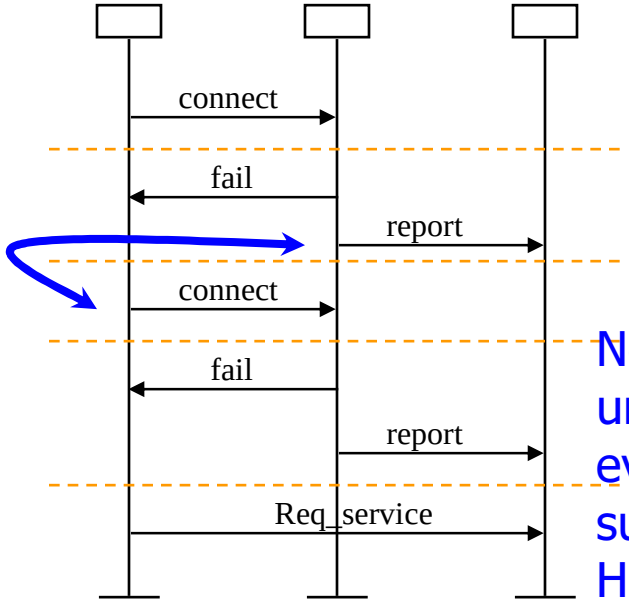
More?

Visual concatenation semantics: extend process lines.

An execution: infinite or maximal



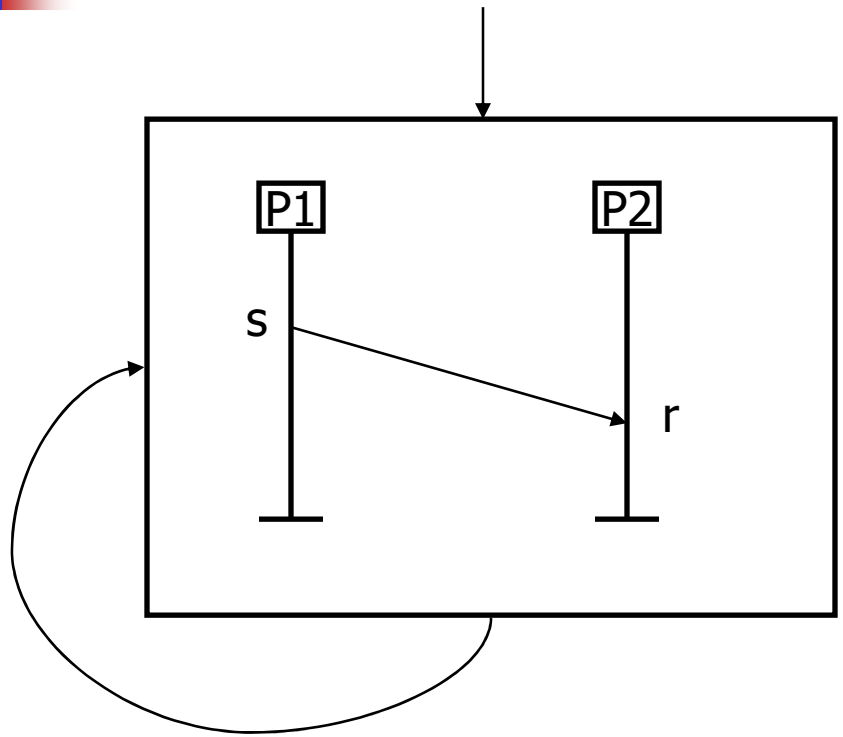
Execution: ***ACACD***



Note: unordered even if in subsequent HMSC boxes!!

... and more.

Star operation over partial order: not necessarily *regular ("recognizable")*

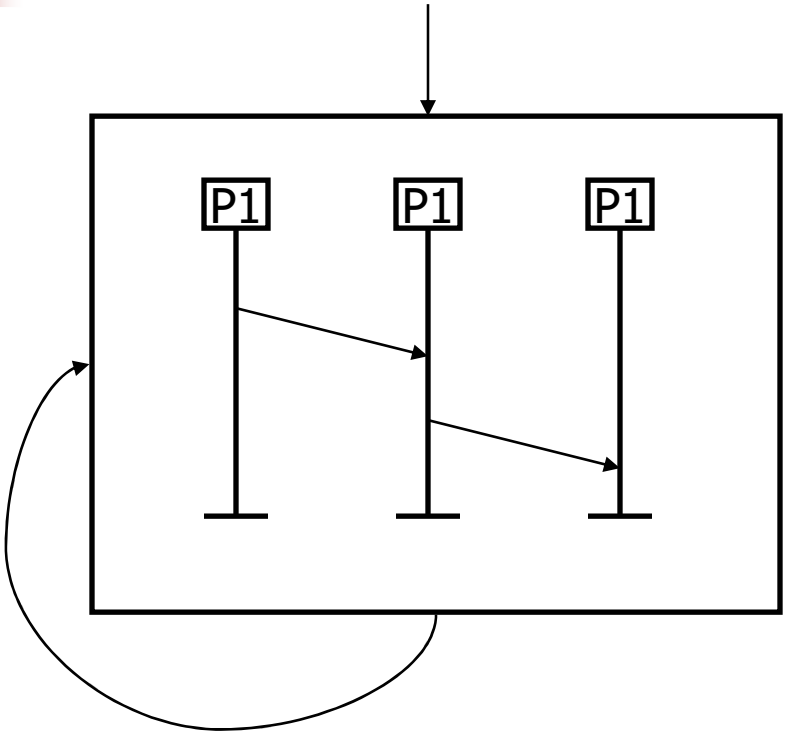


Language:

At least as many send events as receives in every prefix

(Stack Machine Recognizable)

Star operation over partial order: not necessarily *regular ("recognizable")*



Language:

Even worse…

(Not even Stack Machine Recognizable)

Races in HMSCs. Definition

- For each HMSC *M* execution *Ex*, define
  - the linearizations according to the visual order $\text{lin}_{vis}(Ex)$ and
  - the linearizations according to the causal order $\text{lin}_{caus}(Ex)$.
    Extend to all executions: $\text{lin}_{vis}(Ex)$ and $\text{lin}_{caus}(Ex)$.
- Always $\text{lin}_{vis}(Ex) \subseteq \text{lin}_{caus}(Ex)$. (Visual order will *not* allow exchanging two receives on same process!!)
- ***Races*** : <u>when</u> $\text{lin}_{vis}(Ex) \subset \text{lin}_{caus}(Ex)$.

Similar to Mazurkiewicz Traces

(semi commutation: can exchange receive with next send, but not vice versa).

Alphabet {a,b,c}

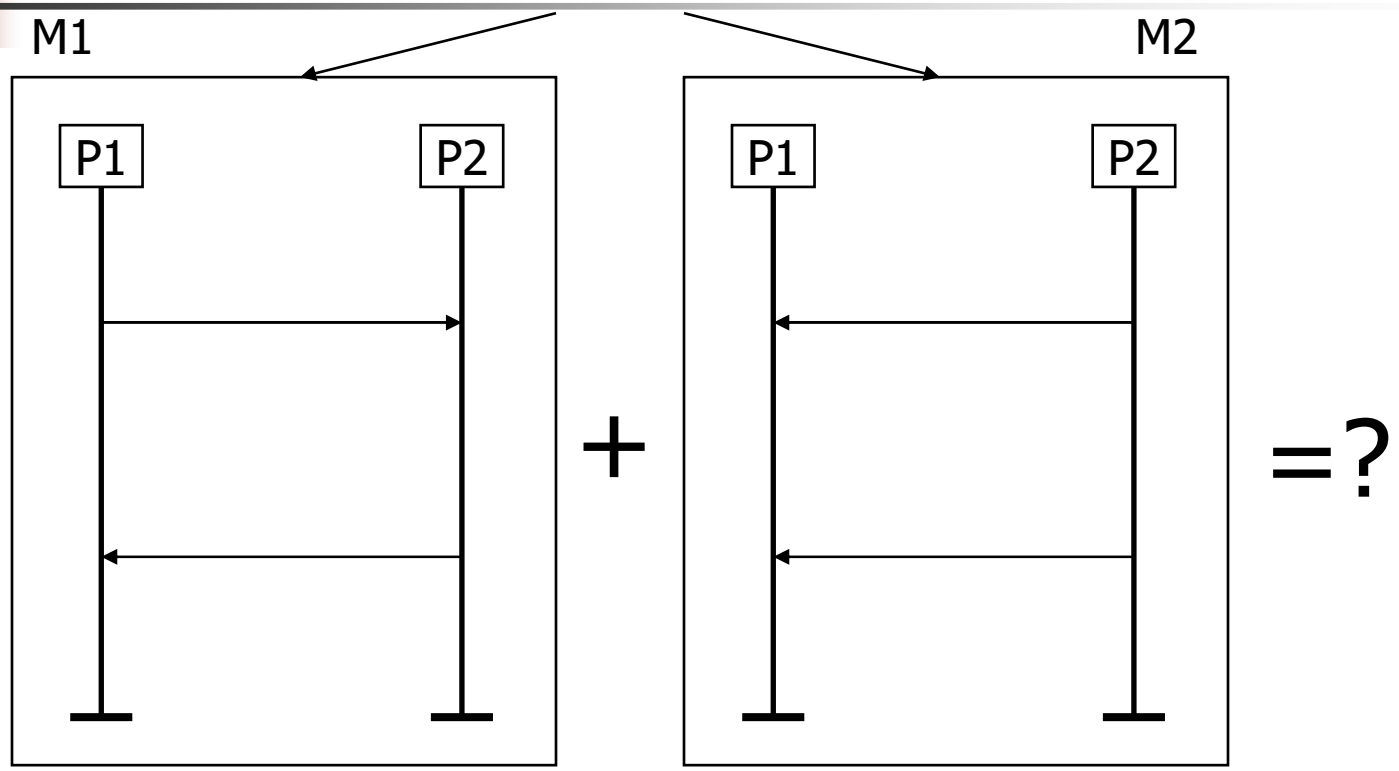
Independence: aIb, bIc

Equivalence classes of words (denoted using representatives):
[aabb]=[abba]

Regular trace language: can be defined using concatenation, star, union, intersection.

Note: [ab]* is not regular (recognizable).

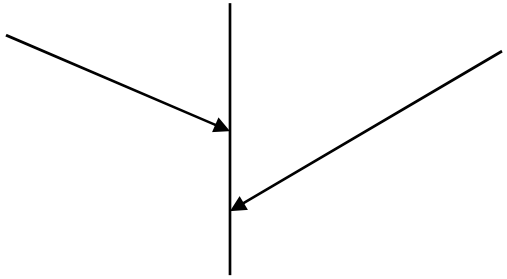
Other problems… Global decision

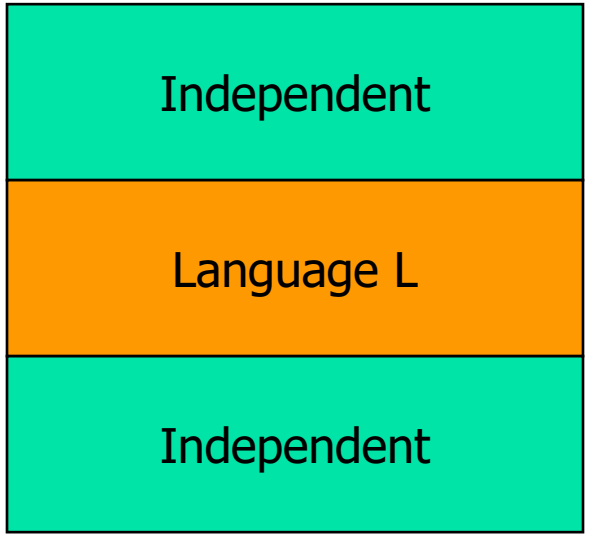


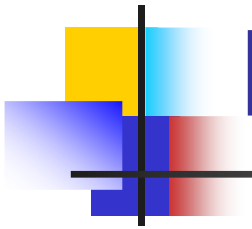
What if one process will start to behave according to M1 and the other will start according to M2?

Races for HMSCs (intuition only)

- Undecidable [MP99]
  - Translate to language theory of *traces*, which are closed w.r.t. commuting certain pairs of letters.
  - Intuition: moving from visual to causal semantic introduces more commutations:
    Two receives on the same process line (from different processes) are **dependent** on visual and **independent** on causal order.
  - Reduction to universality of *trace* languages (things are independent with causal semantics).



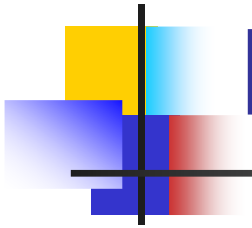




Model checking

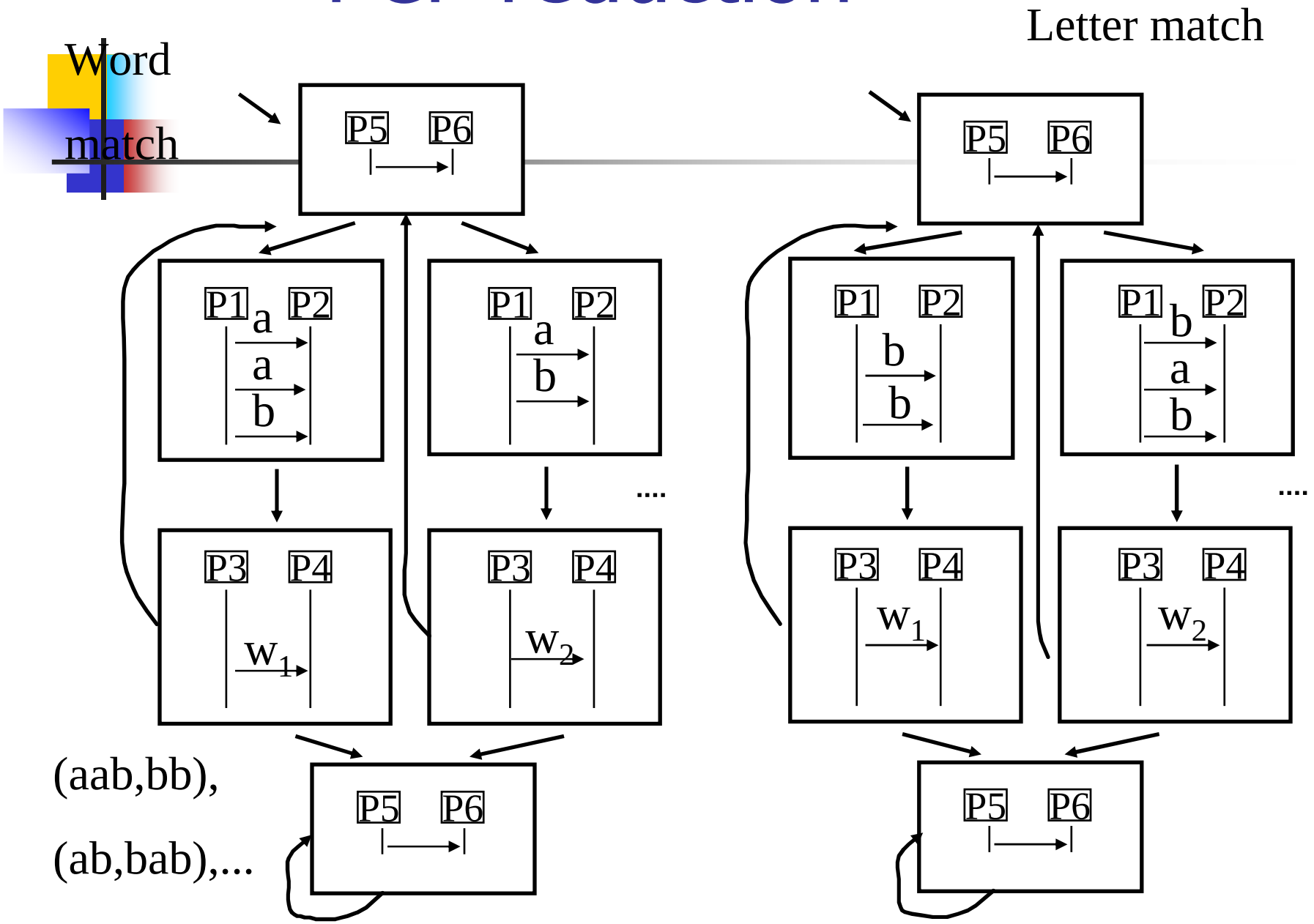
- Write both specification and system as HMSCs. Check emptiness of intersection.
- Write specification in LTL. Interpret over the linearizations of the partial orders.
- In both cases: undecidable.

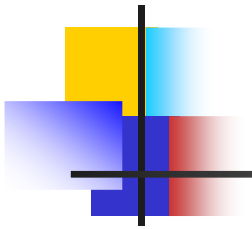
Post Correspondence Problem



- List of pairs:
  w_1:(aab,bb), w_2:(ab,bab), … w_n:(a,bb).
  Want to find if there is a set of indices
  i_1, i_2, …, i_k, such that concatenating
  the lefthand words and concatenating
  the righthand words is the same.
- Supose we take indices 1, 2, n, 1. We get:
- lefthand: aab ab a aab
- righthand: bb bab bb bb

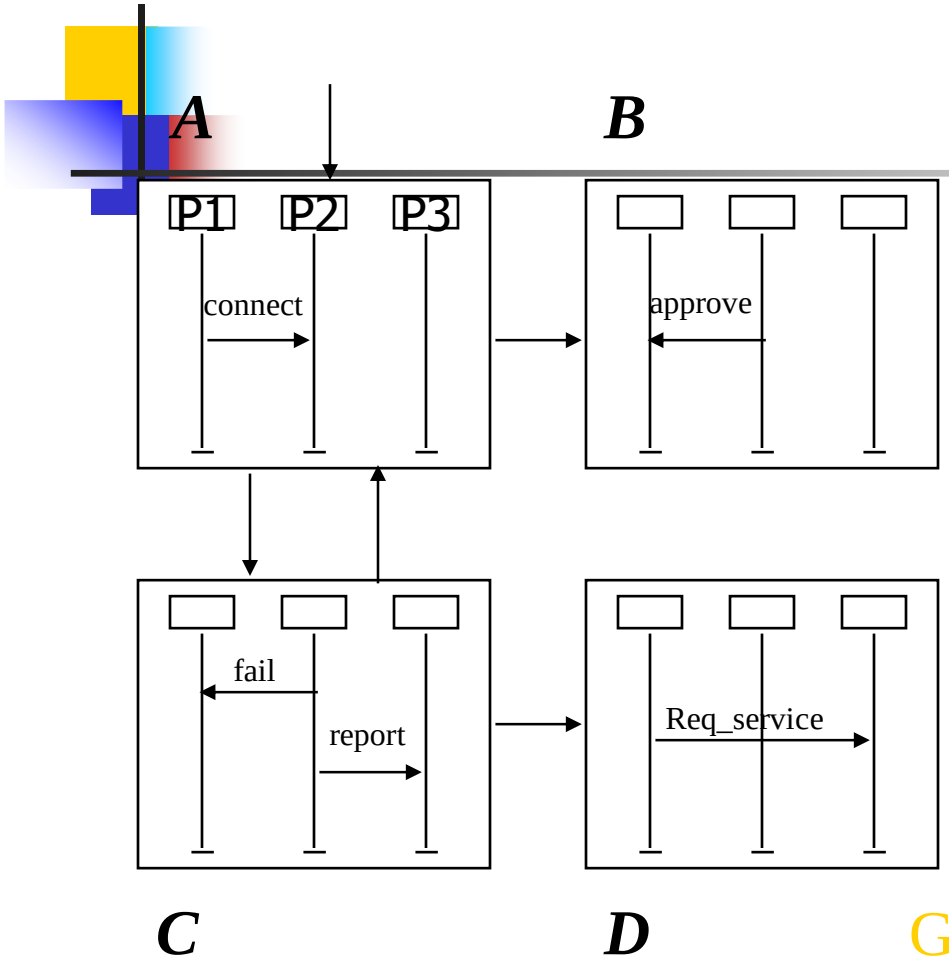
PCP reduction





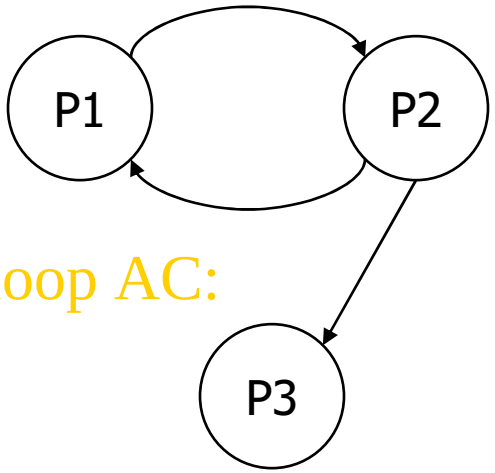
Some solutions:

- The specification HMSCs allows any additional gaps [MPS98].
- Put limit on message queues [Holzmann]
- Strongly connected communication graph for each cycle [AY, MP].
  This also gives a limit (exponential in number of processes) on needed size of message queues.



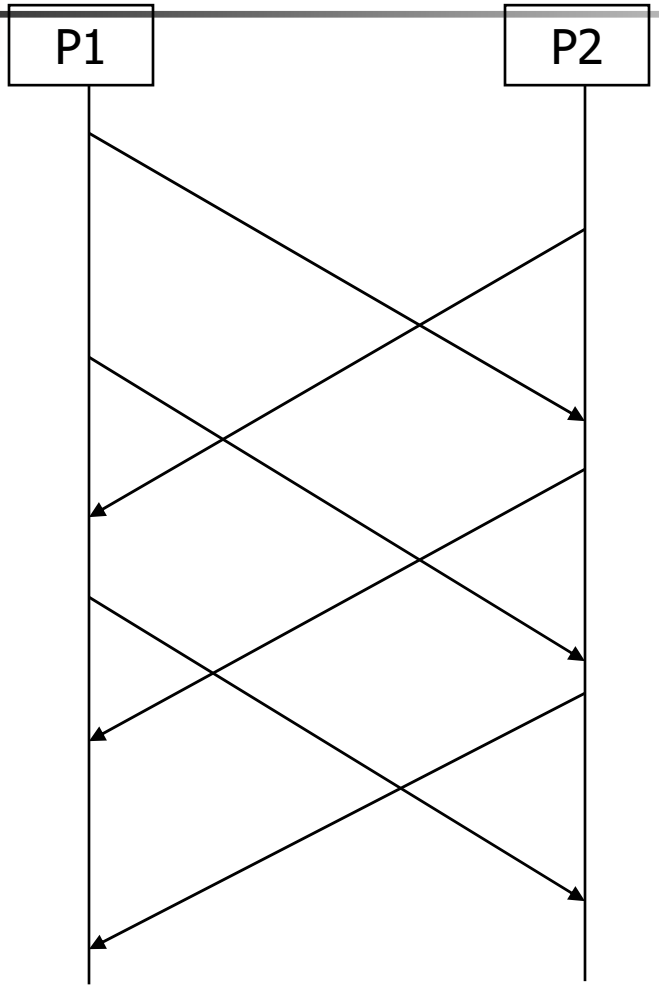
Obtain decidability under the following condition [MP99,AY99]: Every HMSCs cycle covers a strongly connected component in the communication graph: An edge exist from a process Pi to a process Pj if there is a communication from Pi to Pj.

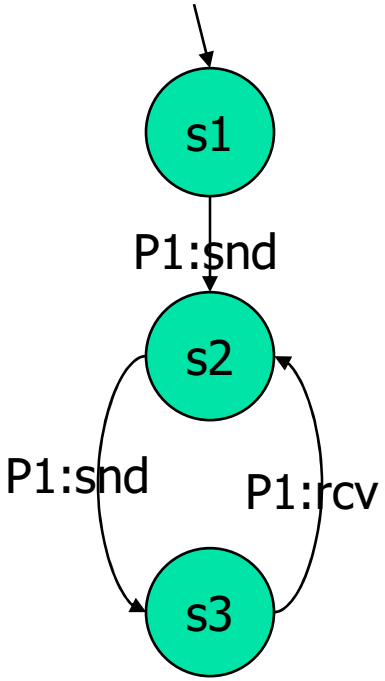
Then system finite and regular!

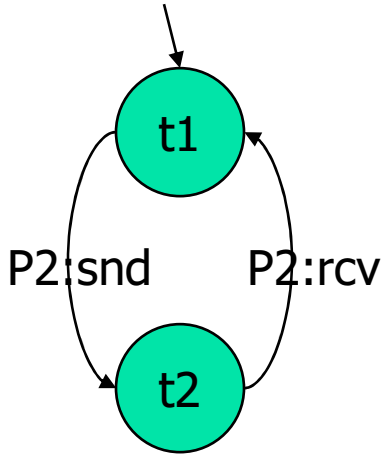


Graph of loop AC:

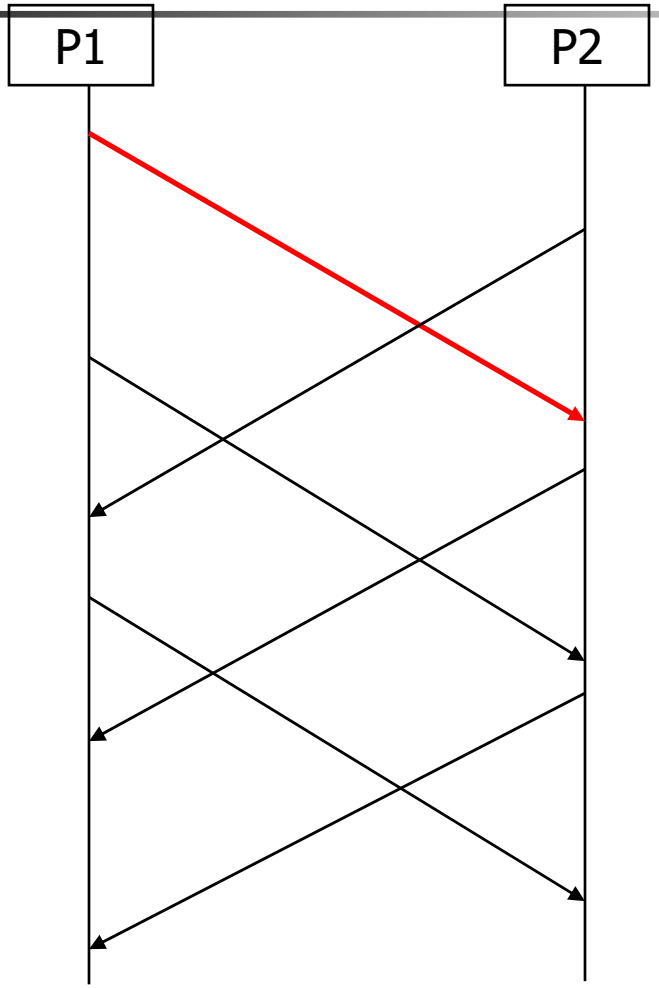
Problem with describing protocols



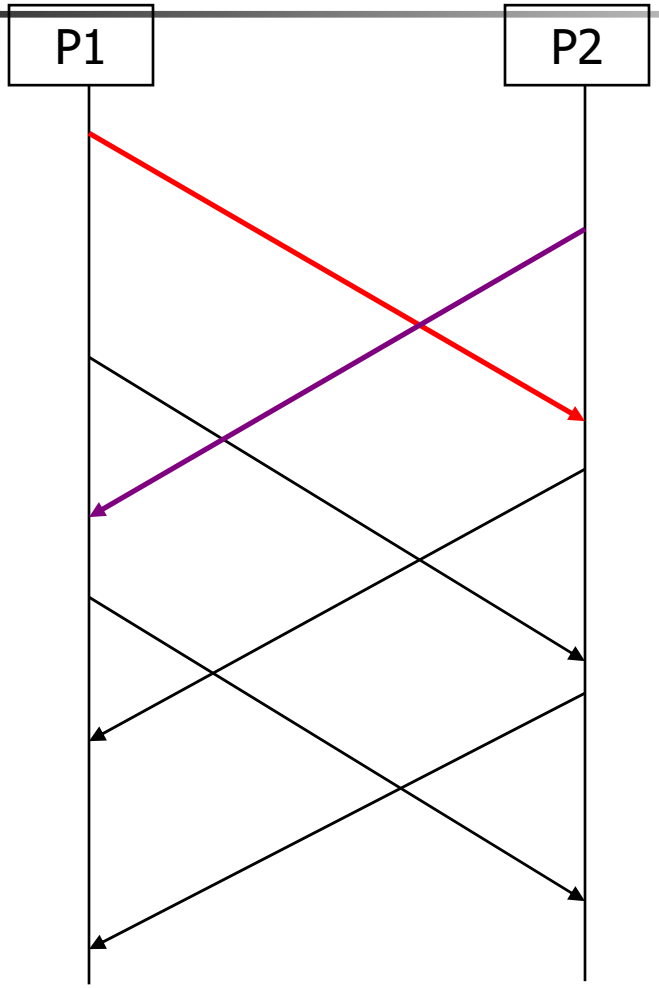




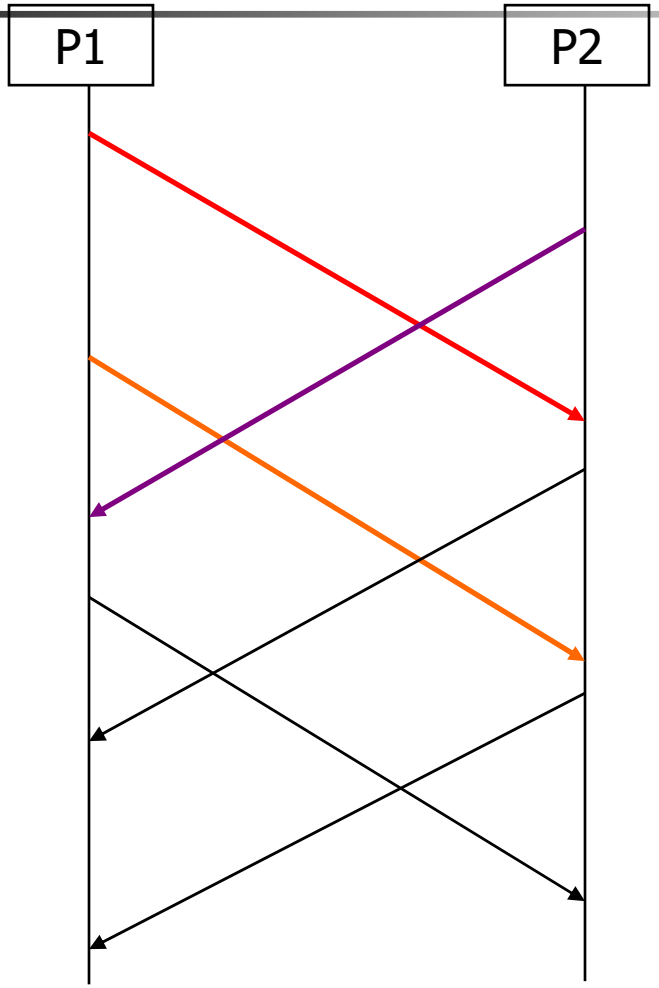
Problem with describing protocols



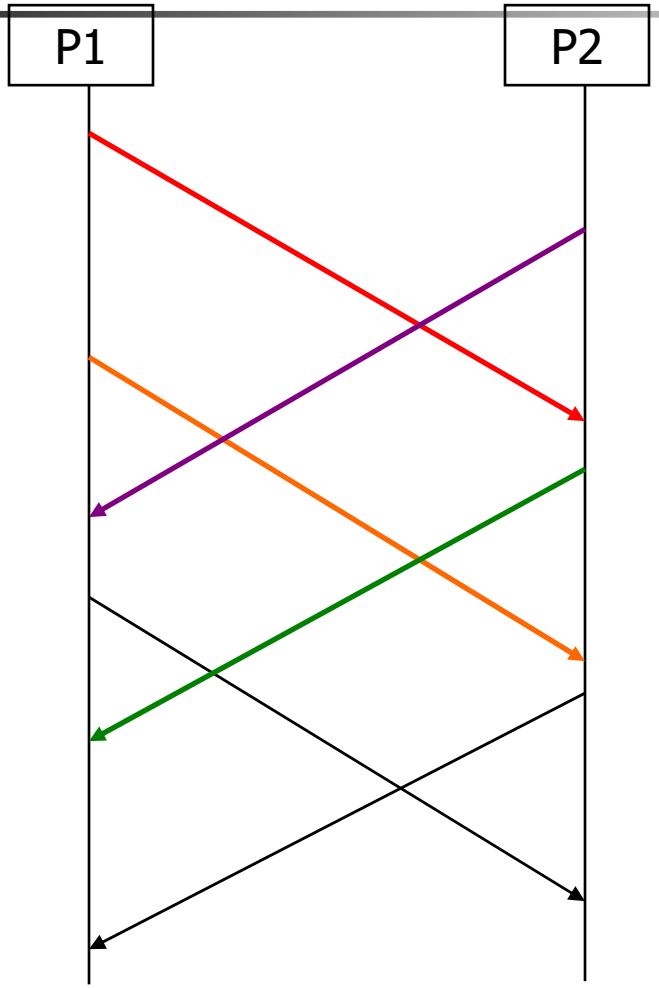
Problem with describing protocols



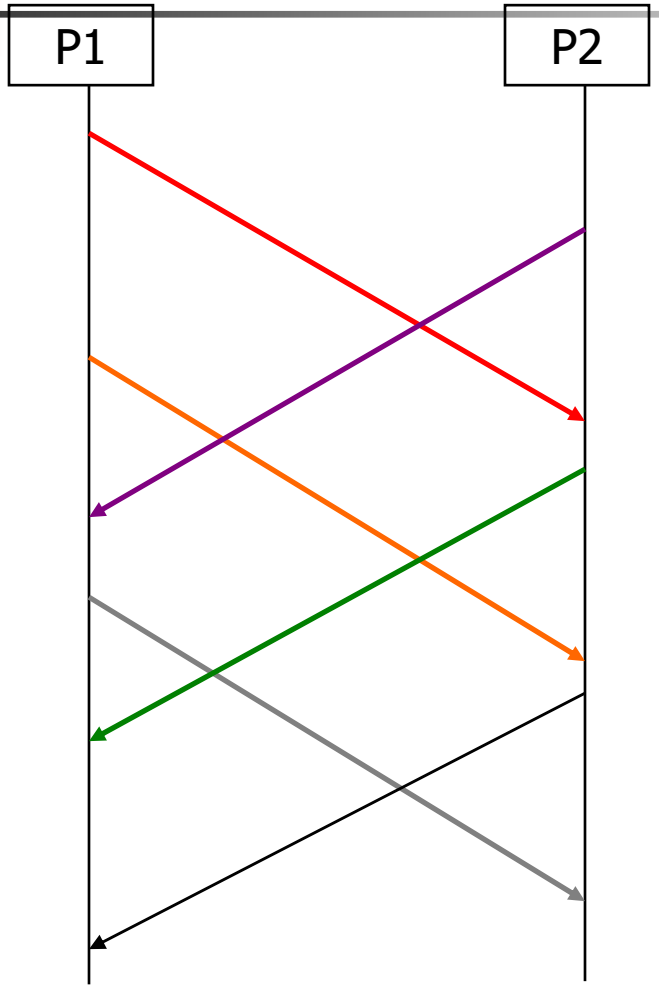
Problem with describing protocols



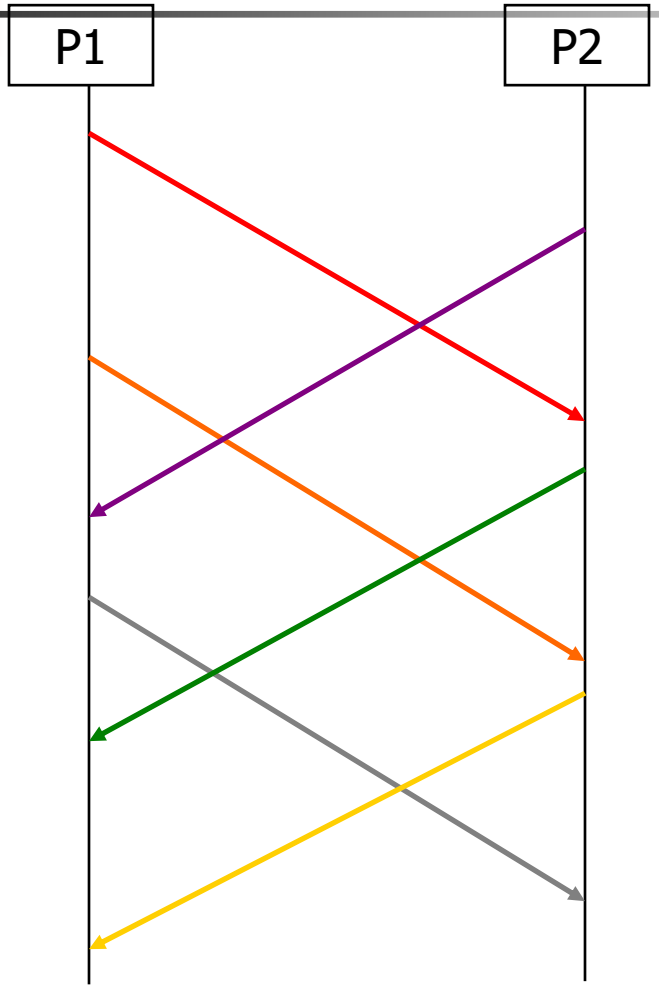
Problem with describing protocols



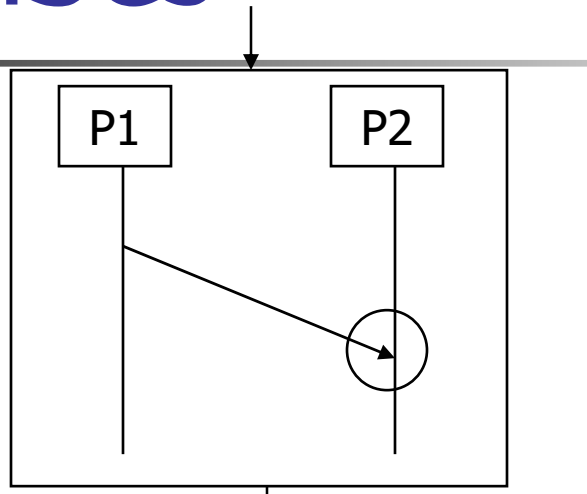
Problem with describing protocols

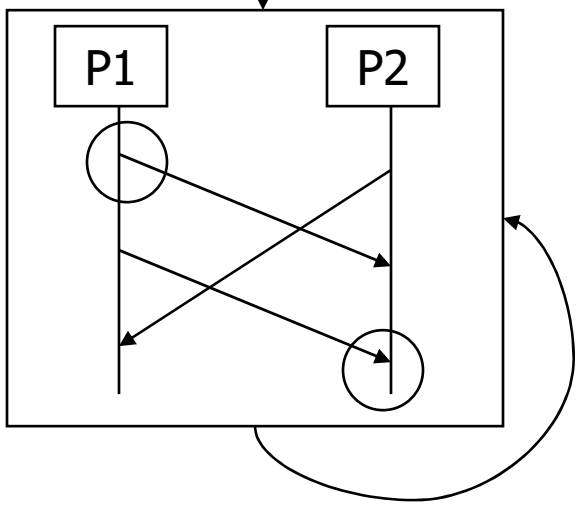


Problem with describing protocols



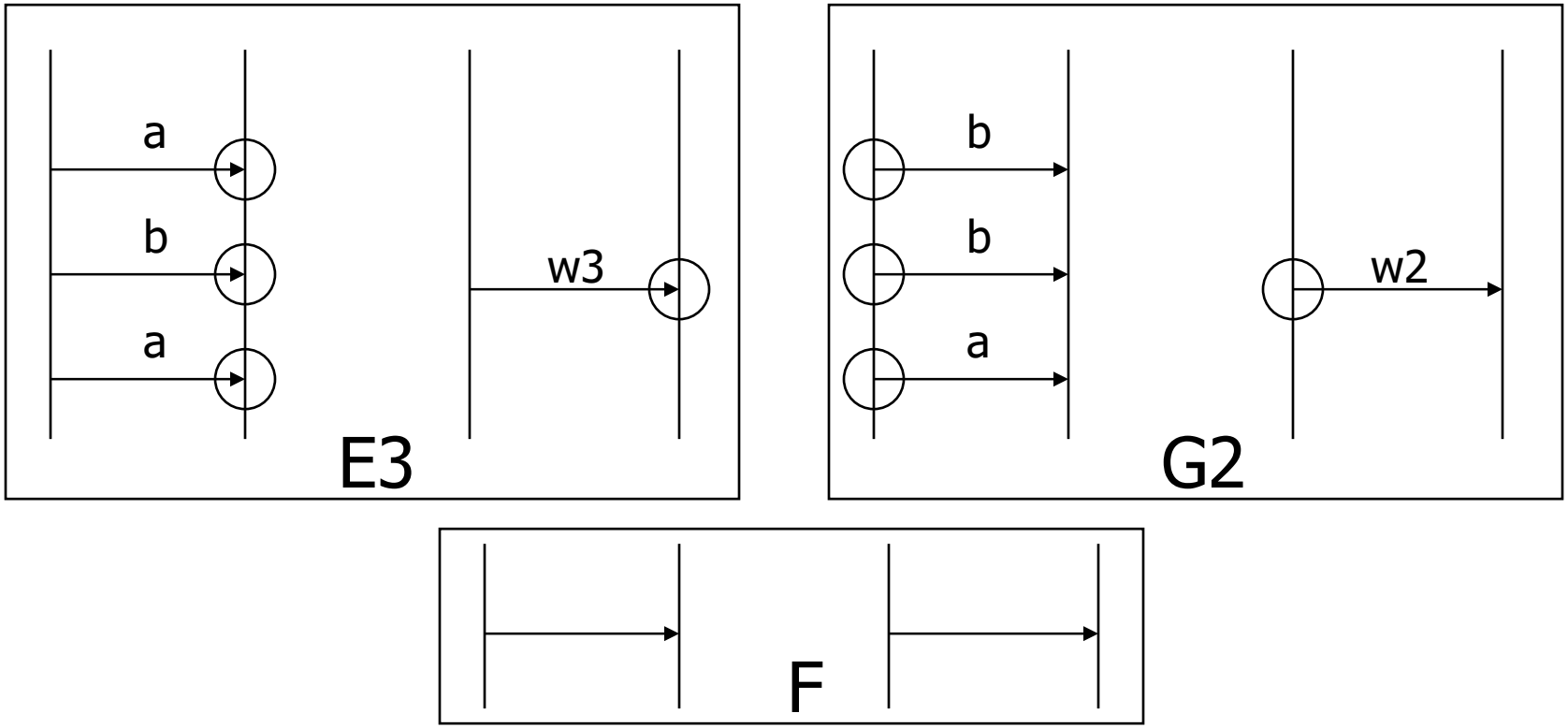
Solution: Compositional HMSCs

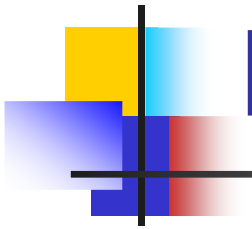




Even emptiness is undecideable!
(but this HCMSC has strange "executions)

$$(E1+E2+\ldots+Em)^{+}\,(G1+G2+\ldots+Gm)^{+}\,F$$





Left closed CHMSCs

- Does not allow unmatched receive event that is not yet matched by a previous unmatched send.
- HCMSC is realizable if every path is matched.
- Can be checked in polynomial time using a nondeterministic stack machine.

How to check for realizability?

- What can go wrong?

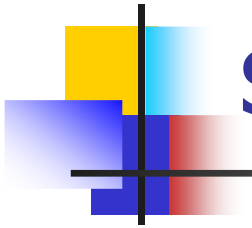
1. More unmatched receives than sends in a prefix.
2. Overtake: the *k*th unmatched send before a matched pair, the *k*th receive after.
3. The *k*th unmatched send has name C, the *k*th unmatched receive has name D.

- How to check with a stack machine *for each pair of processes*?

1+2: Push a £ for each unmatched send (per message name), pop a £ for each unmatched receive.

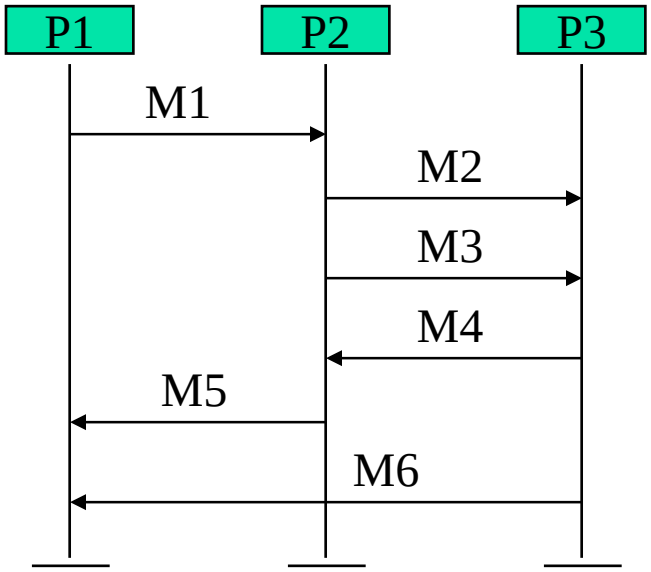
3: Guess that it's a name mismatch upon seeing an unmatched send. Ignore further sends. Pop £ as usual for receives, until corresponding receive occurs.

Now we can translate finite state protocols to CHMSCs



- Any finite state protocol can be translated.
- Trivial translation: any transition in finite state graph makes one CHMSC node, with possibly an unmatched message.
- This does not give more information than finite state graph.
- Try to optimize: take some paths.
  - Break graph into cycle free paths (e.g., using DFS and back arrows).
  - Use partial order reduction (sleep sets) to minimize number of paths.

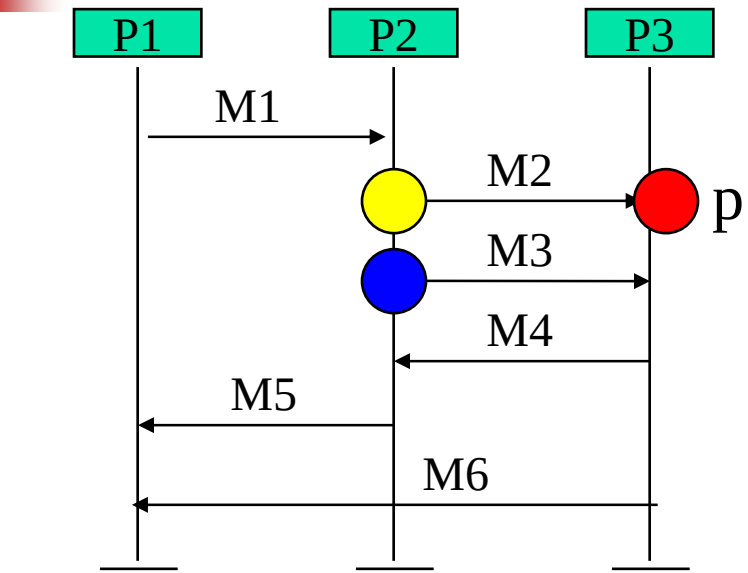
Recall our original example

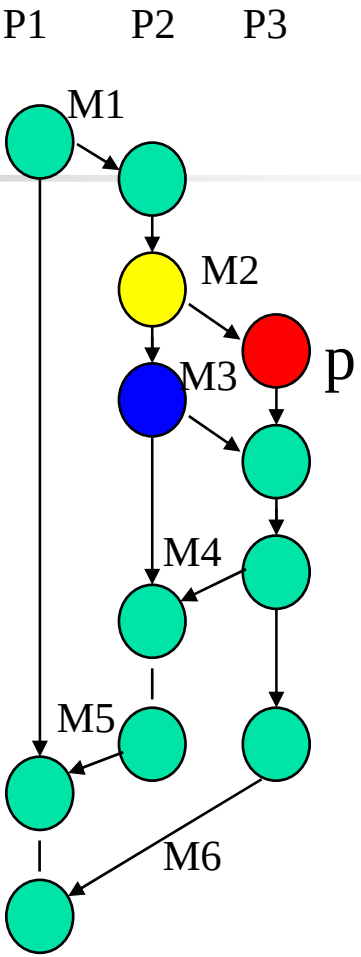


The logic TLC [APP] over MSCs.

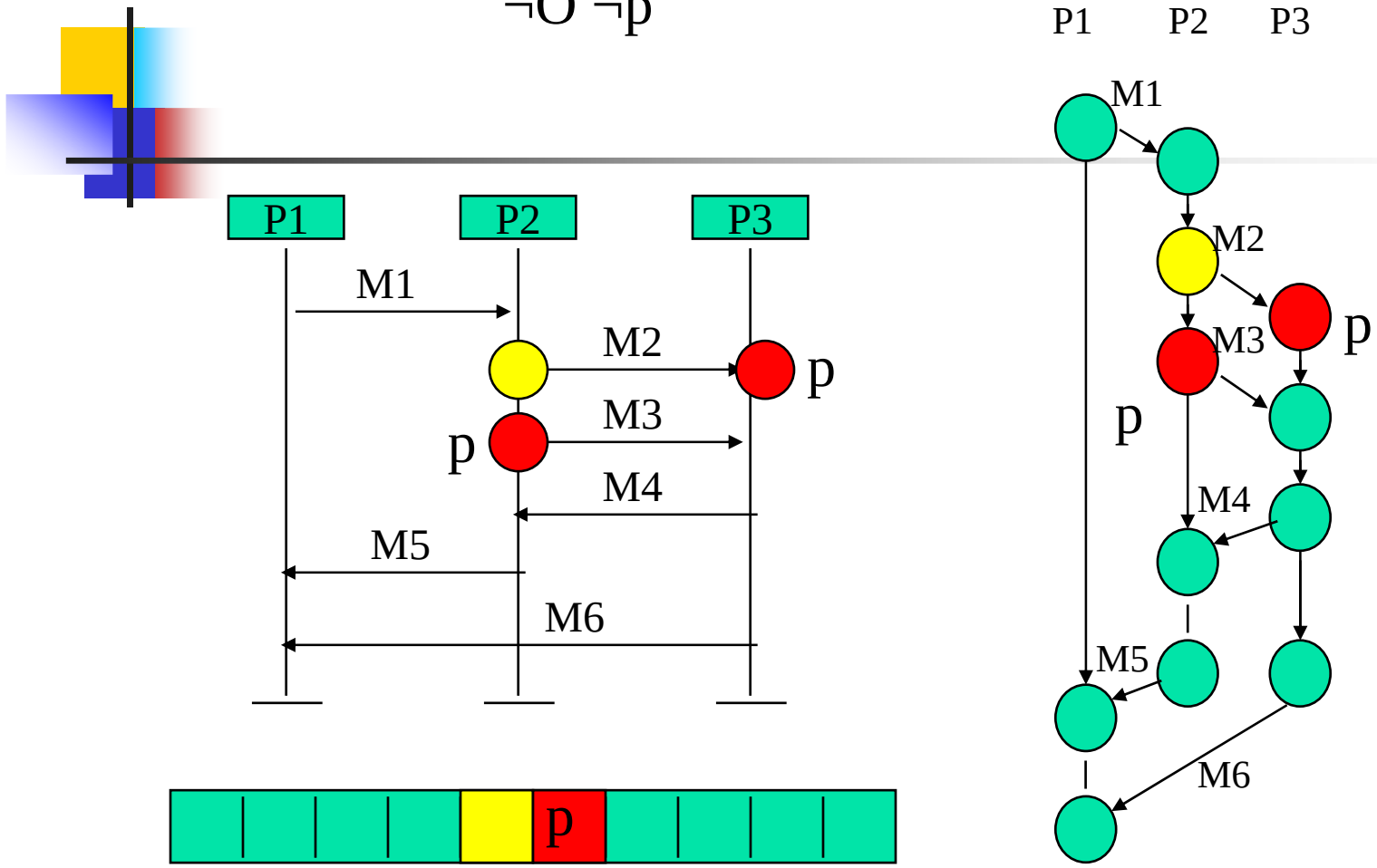
Label events with propositions.

Nexttime: O p

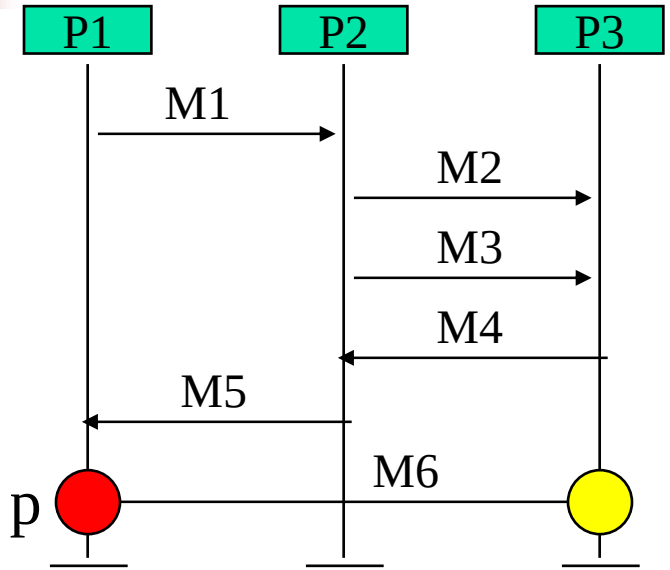


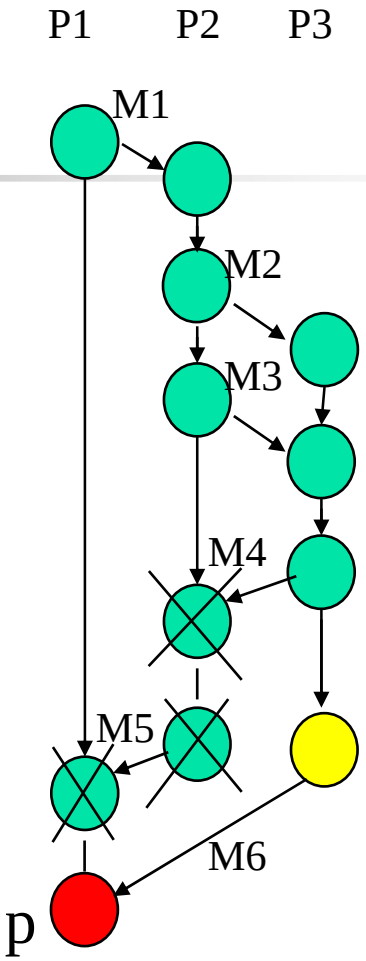


¬O ¬p

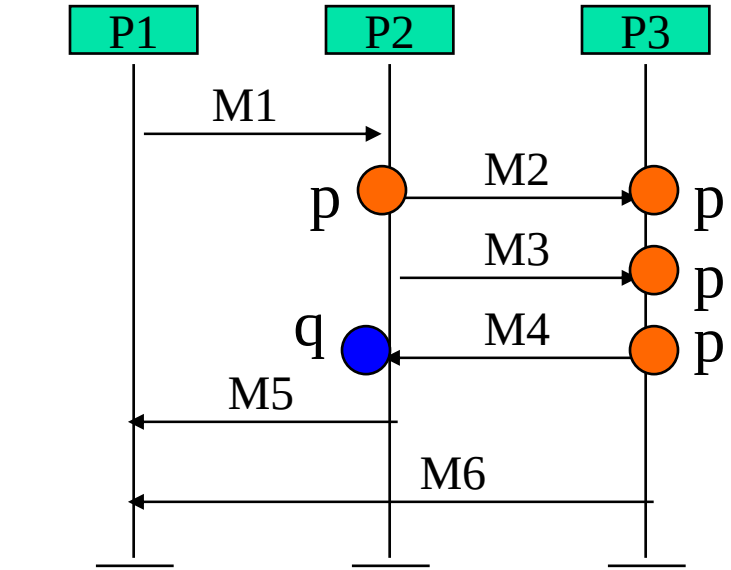


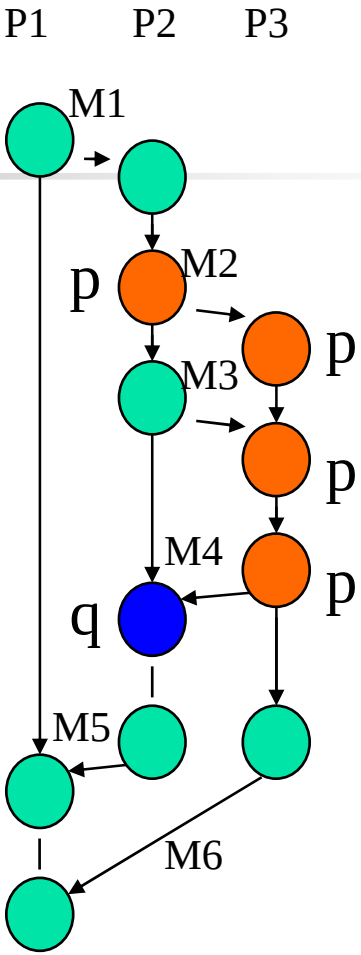
O p





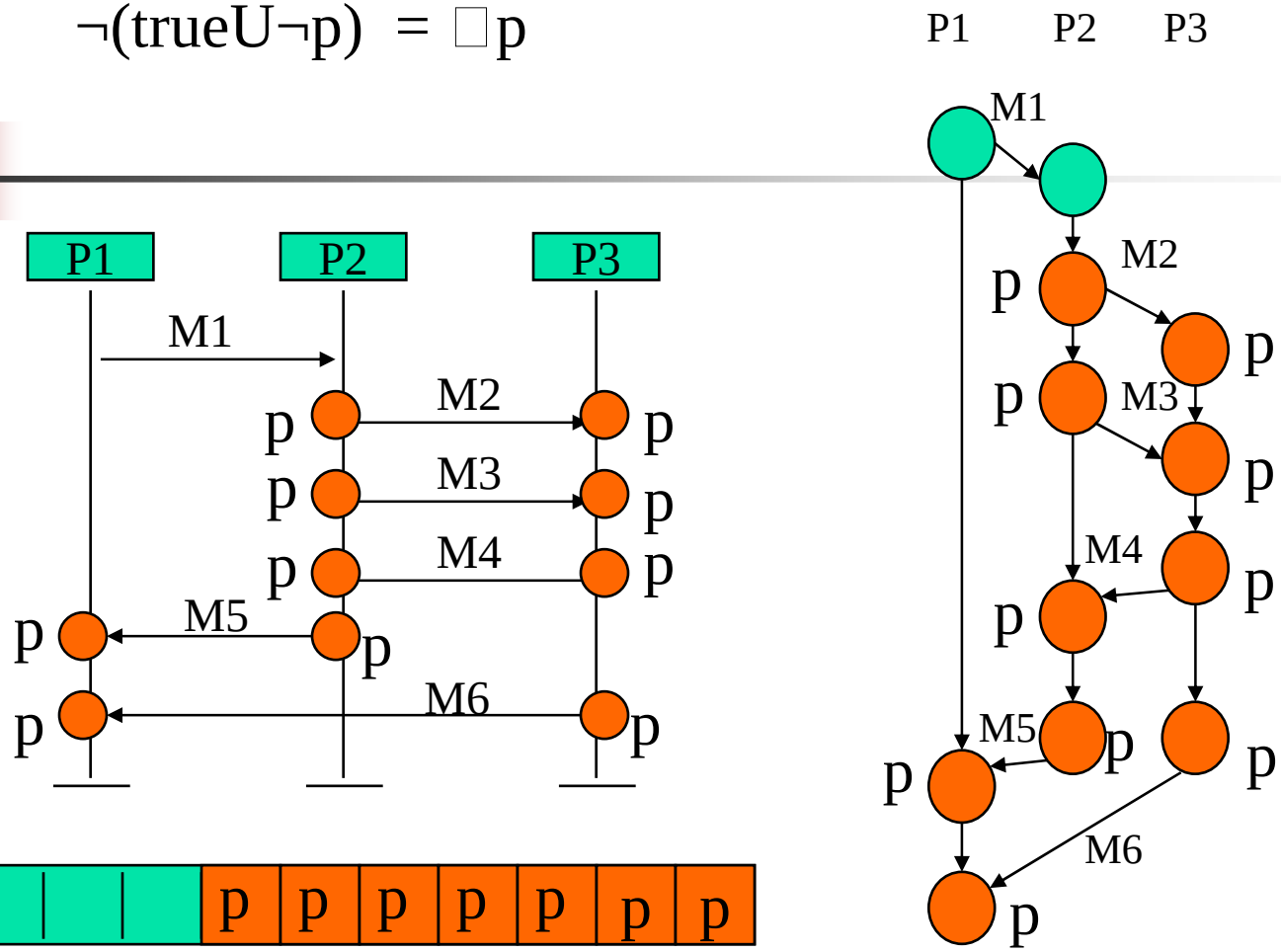
Until: pUq





true U q = <>q

$\neg(\text{true}U\neg p) = \Box p$



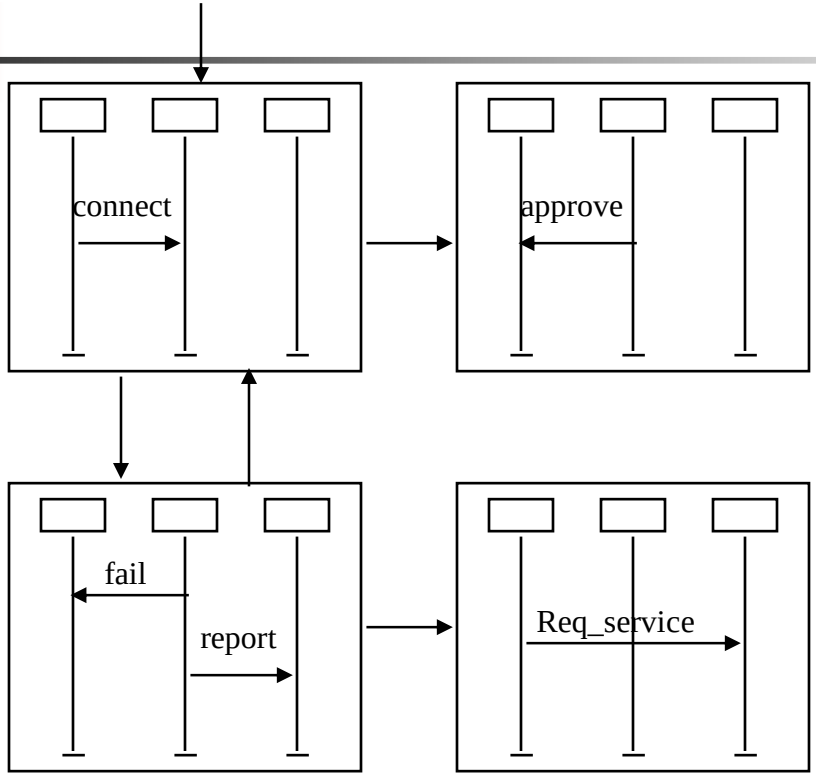
Some specifications

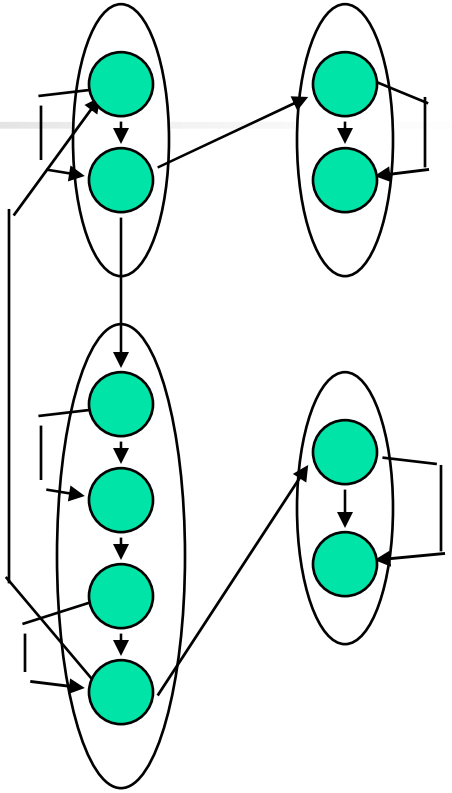
□(req --> <> ack) Every request is followed by acknowledge.

¬<>(transA /\ <> (transB /\ <>transA)) Transaction B cannot interfere with transaction A.

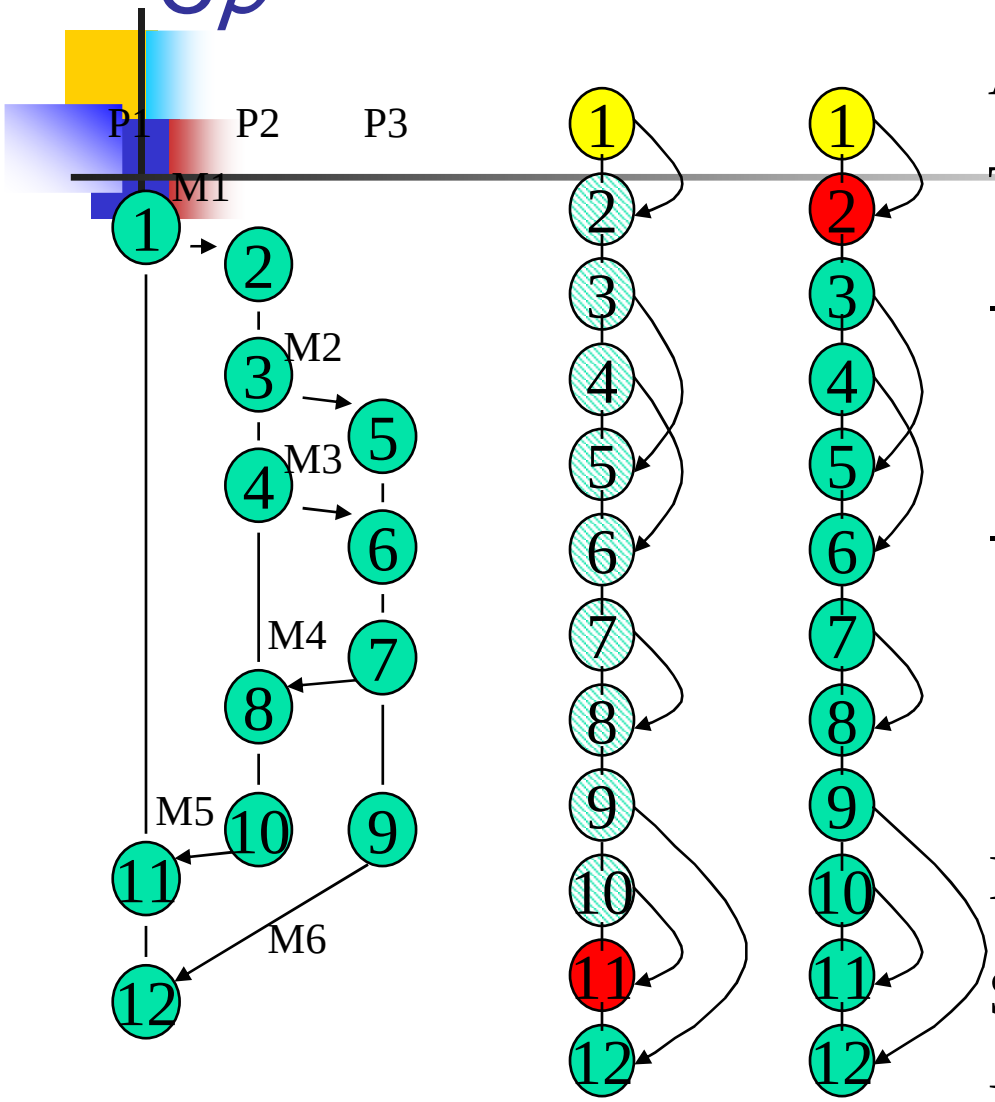
□(beginA --> O (transA U finishA)) The execution of transaction A is not interrupted by any other event.

HMSC linearizations





Intuition behind algorithm for O*p*



Aut. with 2 successors relations.

There are two cases:

- *p* holds for matching receive.

  Then use 2nd successor rel.

- *p* holds for successor in proc.

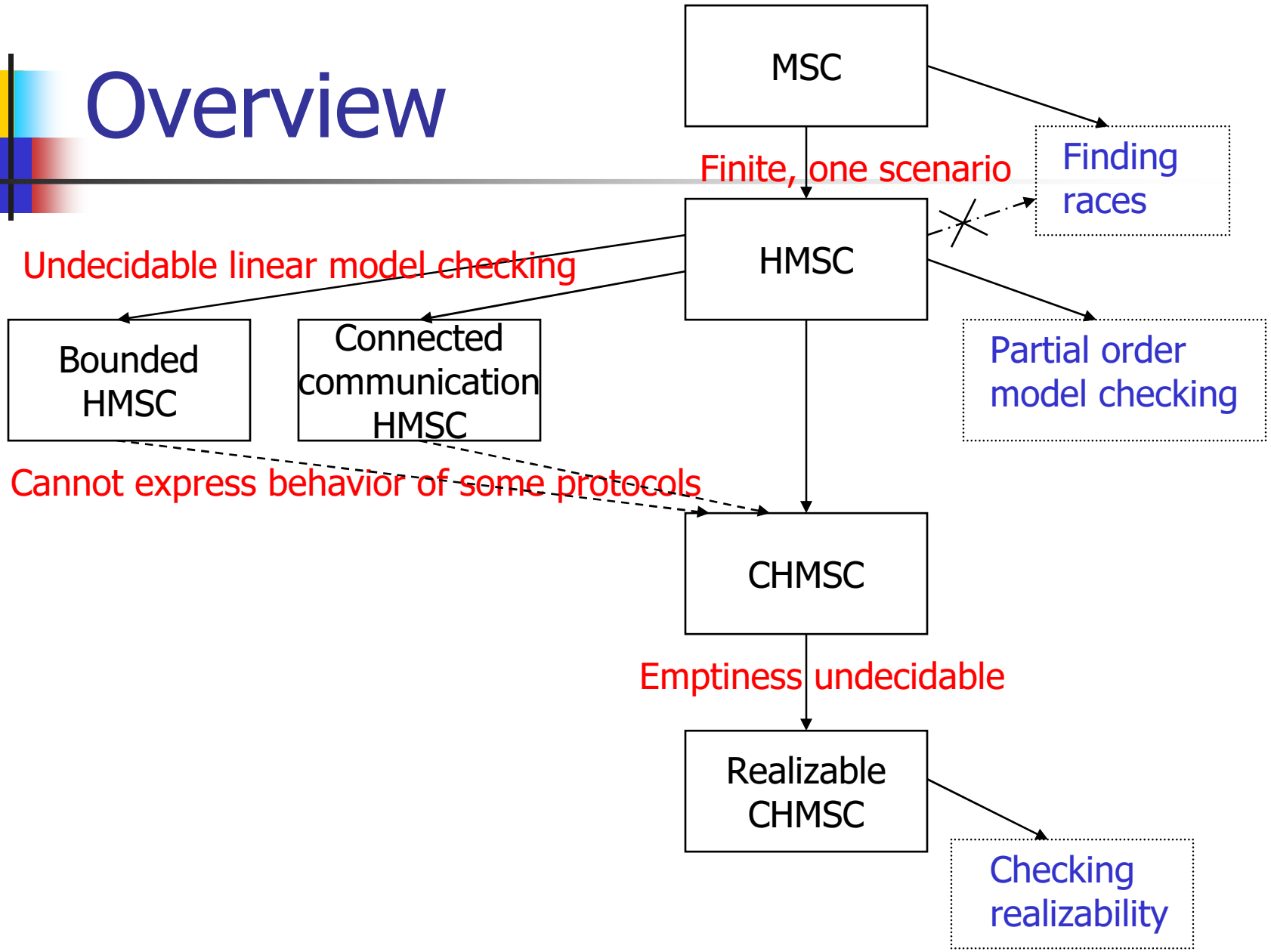
  Then wait to see event of same process.

Intersect:

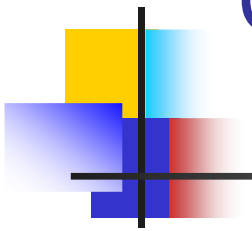
System autom. (linearizations)

Property autom. (of ¬**prop**)

Overview


MSC
Finite, one scenario
Finding races
HMSC
Undecidable linear model checking
Bounded HMSC
Connected communication HMSC
Partial order model checking
Cannot express behavior of some protocols
CHMSC
Emptiness undecidable
Realizable CHMSC
Checking realizability

Conclusions



- Visual notation have advantages over textual representation.
- MSCs is a standard for describing concurrent interactions.
- MSCs are based on partial order semantics.
- MSCs raise many interesting research problems, e.g., race condition.
- Model checking for MSCs is undecidable [GP,AY].
- TLC model checking is based on partial order semantics and is decidable.
- Some extensions to the MSC standard are useful, e.g., CHMSCs, LSCs.