1 REACHABILITY ANALYSIS

- In the following, we will consider certain methods to reduce the size of the reachability graph.
- This means that some of the information will disappear.
- Two methods will be presented:
  1. The symmetry method, and
  2. The stubborn sets.
- Both methods may make the generation of the reachability graph faster but cannot provide a solution to the inherent complexity of the problem (i.e. the state space explosion is always possible).
- Essentially, the methods are heuristics disposing unessential information (w.r.t. the analysis goal) from the reachability graph. Otherwise, this irrelevant information would be stored in the reachability graph.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Reachability analysis

- In the algorithm generating the reachability graph there are two points which increase the computational complexity:
  - On line 6, the **for** loop, and
  - On line 8, the test $M' \notin G$.

Reachability-Graph$(N :< S, T; F, K, M_0, W >)$

```
 1  G : graph; Marking : stack;
 2  G ← ∅;
 3  Markings ← M_0;
 4  while Markings ≠ ∅
 5  do M ← pop(Markings);
 6      for t in enabled(M)
 7      do M' ← fire(t, M);
 8          if M' ∉ G
 9              then push(M', Markings);
10          G ← G ∪ {< M, t, M' >};
```

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

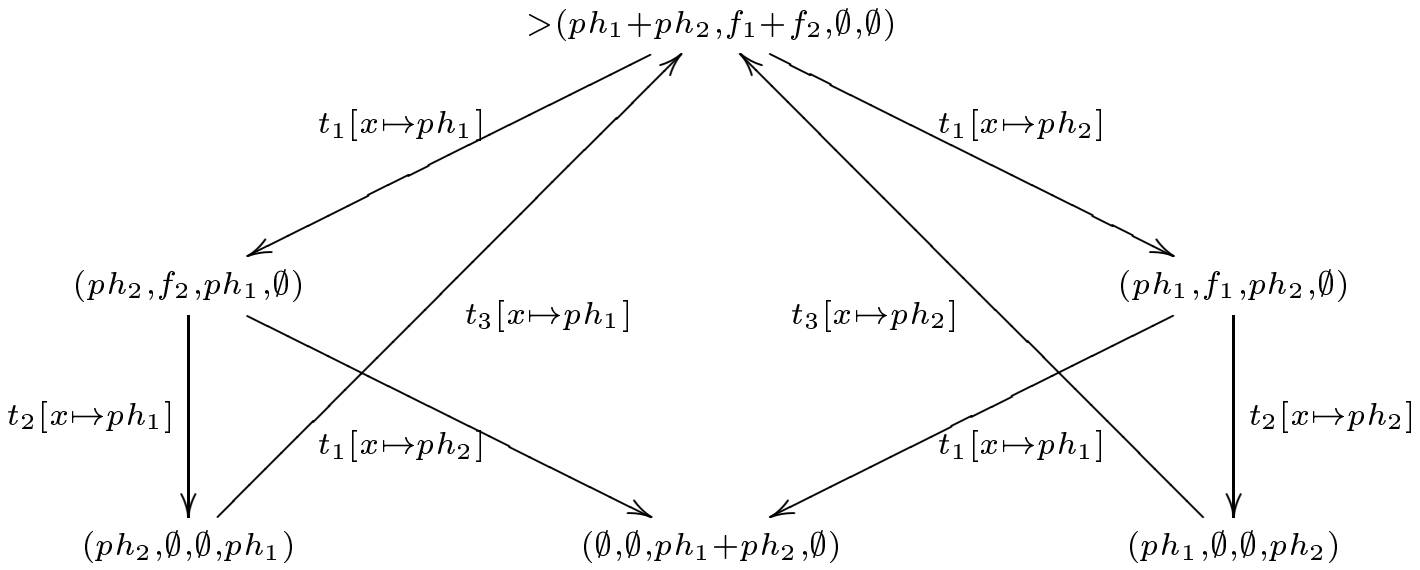
Reachability analysis

- The symmetry method tries to reduce the size of the reachability graph by storing equivalence classes of markings only, resulting in a smaller size of the set of markings to be stored.
- The stubborn set method tries to reduce the reachability graph by firing on line 6 $t \in enabled(M)$ only those transitions which belong to the set $stubborn(M) \subseteq enabled(M)$. This reduces branching in the reachability graph, thus diminishing the number of markings.
- In both methods, however, a trade-off between space and time occurs because both the calculation of symmetric markings and stubborn transition sets may suffer from time complexity.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Symmetry method

- Consider the reachability graph given below:



- Clearly, the graph is symmetric w.r.t. the line between the initial marking and the deadlock.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Symmetry method

- The *symmetry* of a P/T net $\sigma : S \cup T \to S \cup T$ is an *automorphism* (i.e., the P/T net is isomorphic with itself) respecting arc weights:

$$W(< a, a' >) = W(< \sigma(a), \sigma(a') >)$$

  and capacities.
- A symmetry σ can be extended to a mapping of markings $\sigma : (S \to \mathbb{N}) \to (\mathbb{S} \to \mathbb{N})$ by defining:

$$(\sigma(M))(s) = M(\sigma^{-1}(s))$$

- Hence, we can easily deduce that

$$M[t > M' \equiv \sigma(M)[\sigma(t) > \sigma(M')$$

  for all $M, M' \in \mathbb{M}, t \in T$.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Symmetry method

- A set of symmetries Σ, for which the following holds:
  1. $id \in \Sigma$, where $id(M) = M$,
  2. $\sigma \in \Sigma \Rightarrow \sigma^{-1} \in \Sigma$, and
  3. $\sigma, \sigma' \in \Sigma \Rightarrow \sigma' \circ \sigma \in \Sigma$

  constitute a *symmetry group*.
- A symmetry group defines an equivalence relation for markings:

$$M \equiv M' \stackrel{\text{def}}{=} (\exists \sigma \in \Sigma : M = \sigma(M'))$$

- The marking M is called *symmetric* iff

$$\forall \sigma \in \Sigma : M = \sigma(M)$$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Symmetry method

SYMMETRIC-RG($N : P/T - Net, S : SymmetryGroup$)

```
 1  G : graph; Marking : stack;
 2  G ← ∅;
 3  Markings ← M_0;
 4  while Markings ≠ ∅
 5  do M ← pop(Markings);
 6      for t in enabled(M)
 7      do M' ← fire(t, M);
 8          if ∃M'' ∈ G ∃σ ∈ S : M'' = σ(M')
 9              then
10                  G ← G ∪ {< M, t, M'' >};
11              else
12                  G ← G ∪ {< M, t, M' >};
13                  push(M', Markings);
```

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Symmetry method

- If the initial marking is symmetric the method retains reachable markings:

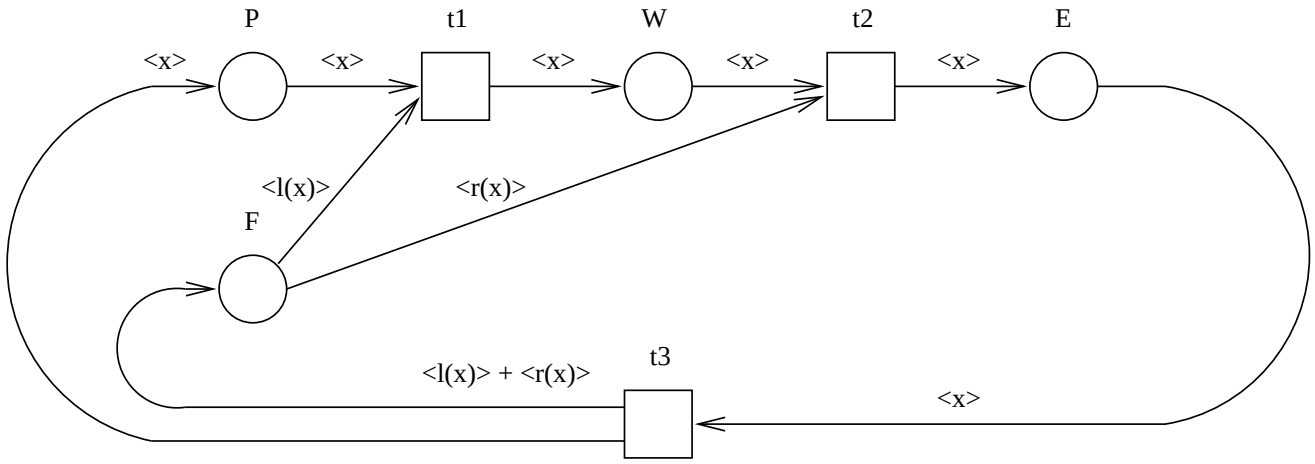
$$M_0[> M \Leftrightarrow \exists M'(M_0[> M' \wedge M' \cong M)$$

- The liveness of a single marking does not necessarily prevail.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Symmetry method

- Consider the two dining philosophers example:



- Initial marking: $P \mapsto \{\langle ph_1 \rangle + \langle ph_2 \rangle\}, F \mapsto \{\langle f_1 \rangle + \langle f_2 \rangle\}$
- Functions: $l(ph_i) = f_i$, $r(ph_i) = f_{(i \bmod 2)+1}$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

- One of the symmetries is as follows:

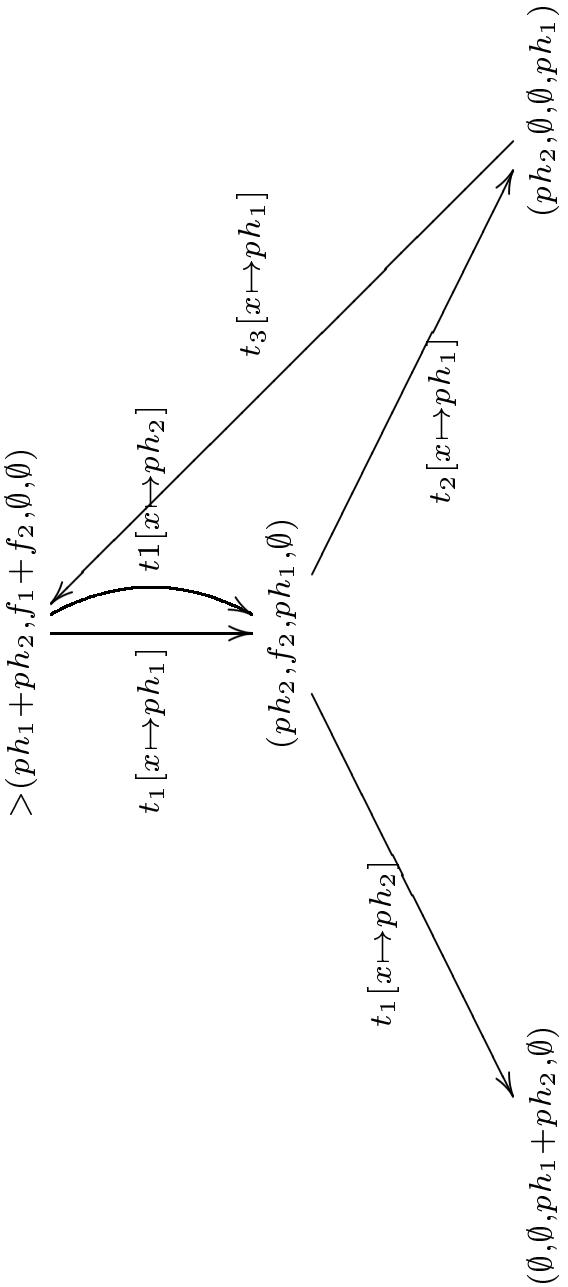
$$\sigma = \begin{pmatrix} P(ph_1) & \mapsto & P(ph_2) \\ P(ph_2) & \mapsto & P(ph_1) \\ F(f_1) & \mapsto & F(f_2) \\ F(f_2) & \mapsto & F(f_1) \\ W(ph_1) & \mapsto & W(ph_2) \\ W(ph_2) & \mapsto & W(ph_1) \\ E(ph_1) & \mapsto & E(ph_2) \\ E(ph_2) & \mapsto & E(ph_1) \end{pmatrix}$$

- Actually, this symmetry *generates* all the other symmetries. E.g., $id = \sigma \circ \sigma$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

SYMMETRY METHOD

- Reduced reachability graph:



©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Symmetry method

- It is clear that the above algorith reduces the size of the reachability graph.
- Surprisingly, the algorithm often is faster than the traditional one.
- Even though in each iteration symmetries are processed, entire subtrees of the reachability graph are left unexplored.
- The bottle neck of the method is in finding the symmetries:
  - User provides the symmetries, or
  - The tool itself computes the symmetries. N.B.: This is a time consuming process. Essentially, it is a question of computing the automorphism of the graph. This is an NP complete problem.
- With `PROD`, it is possible to generate a symmetrically reduced reachability graph. `PROD` computes the symmetries itself.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Stubborn sets

- The underlying idea with the stubborn set method is that on line 6 we try to consider subsets of the set of transitions which are in concession, thus reducing branching in the reachability graph.
- In particular, we try to split the set of transition T into two sets T_s and $T \backslash T_s$ such that none of the transitions in T_s prevent any the of the transitions in $T \backslash T_s$ from firing.

  Assuming that the diagram below is commuting:

$$\begin{array}{ccc} M & \xrightarrow{\sigma} & M' \\ \downarrow t & & \downarrow t \\ M'' & \xrightarrow{\sigma} & M''' \end{array}$$

  where $t \in T_s$ and for all $t' \in \sigma : t' \in T \backslash T_s$.

  W.r.t. such a property which holds in the marking M''' it is 'negligible' if it is found on the path $t\sigma$ or σt.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Stubborn sets

Definition 1 Let N be a P/T net and M one of its markings. Now a set of transitions T_s is *stubborn* in the marking M iff:

$$T_s \subseteq T \wedge (\exists t \in T_s \wedge M[t>),$$

$$\forall t \in T_s \quad : \quad M[t> \Rightarrow ({}^\bullet t)^\bullet \subseteq T_s,$$

$$\forall t \in T_s \quad : \quad \neg M[t> \Rightarrow (\exists s \in {}^\bullet t \wedge M(s) \leq W(s,t) \wedge {}^\bullet s \subseteq T_s)$$

- The set of all transitions T is trivially stubborn.
- Starting from a marking M, the stubborn set can be computed (inefficiently) as follows:
  1. Select a transition t which is in concession in the marking M. Let $T_s = \{t\}$.
  2. Select a transition $t' \in T_s$ (belonging to the transitions which are still unexplored):
     (a) $M[t>$: Condition 2 holds with the assignment $T_s = T_s \cup ({}^\bullet t)^\bullet$. Go to 2.
     (b) $\neg M[t>$: According to Condition 3 find a place s which causes t not to be in concession, and assign $T_s = T_s \cup {}^\bullet s$. Go to 2.
- The algorithm terminates when the set T_s does not change during an iteration.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Stubborn sets

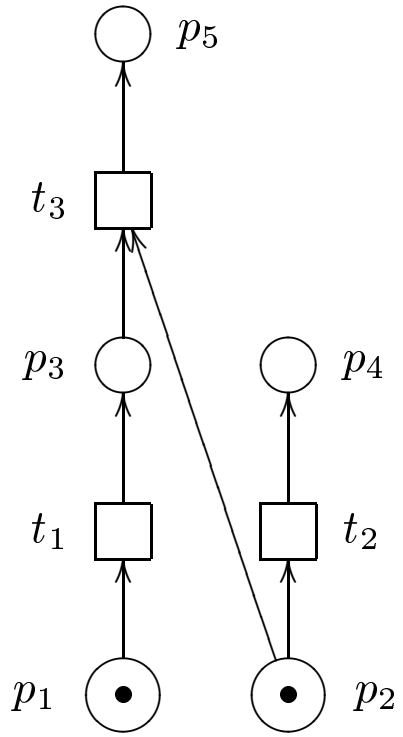
Stubborn-Reachability-Graph($N :< S, T; F, K, M_0, W >$)

```
 1  G : graph; Marking : stack;
 2  G ← ∅;
 3  Markings ← M_0;
 4  while Markings ≠ ∅
 5  do M ← pop(Markings);
 6      for t in stubborn(M)
 7      do M' ← fire(t, M);
 8          if M' ∉ G
 9             then push(M', Markings);
10          G ← G ∪ {< M, t, M' >};
```

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Stubborn sets

- Example



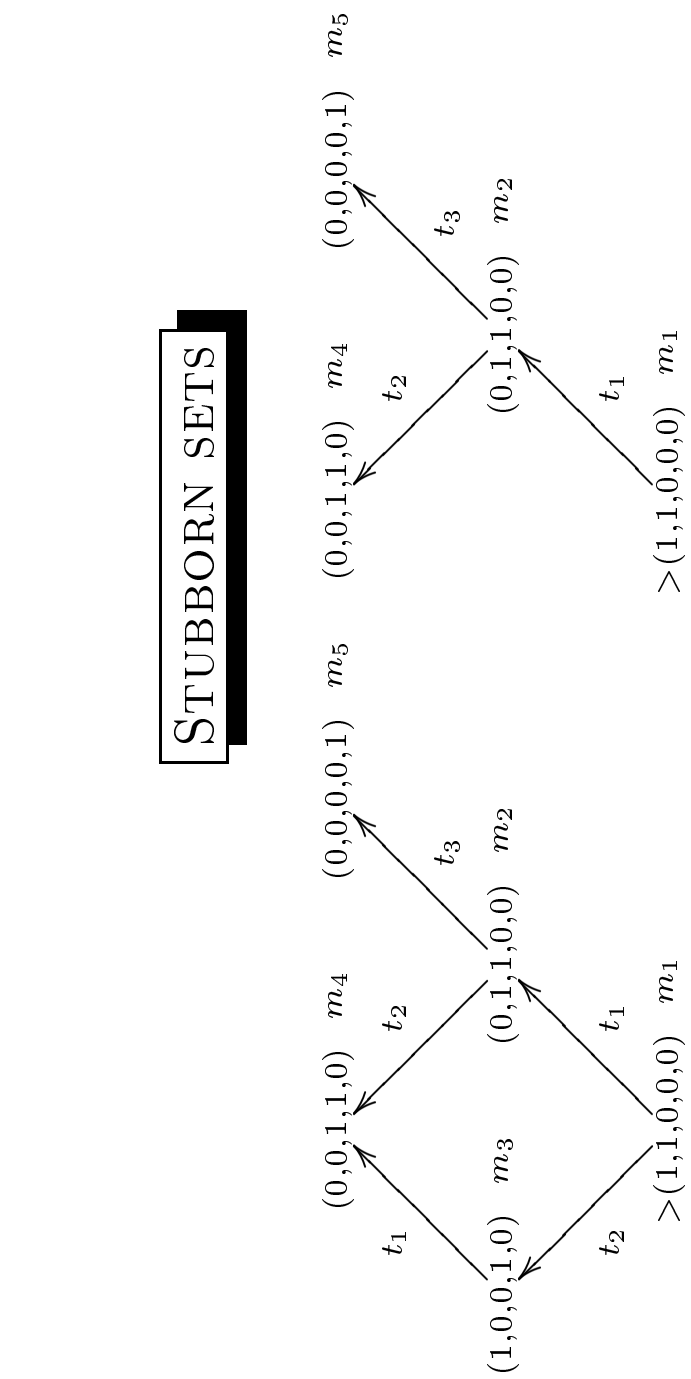
  - Picking t_2:

$$M[t_2 > \quad \Rightarrow \quad ({}^\bullet t_2)^\bullet = \{t_2, t_3\} \subseteq T_s$$
$$\neg M[t_3 > \quad \Rightarrow \quad {}^\bullet(p_3) = \{t_1\} \subseteq T_s$$

    hence, we obtain, $T_s = \{t_1, t_2, t_3\} = T$.
  - Picking t_1, Condition 2 gives us $T_s = \{t_1\}$

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

STUBBORN SETS



©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Stubborn sets

- The size of a stubborn set is largely dependent on which enabled transition is where the computation is started from.
- Additionally, the choice the place s in Condition 3 influences the size of the stubborn set.
- The smallest possible stubborn set is constructed by employing the *deletion algorithm*. Starting from the set of all transitions, one transition at a time is deleted until the set is stubborn again.
- The so called *incremental* algorithm presented above can be improved by taking advantage of Tarjan's algorithm for strongly connected components.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000

Stubborn sets

- The basic version preserves all deadlocks in the system.
- There exists a modification of the stubborn set method for which it is possible to prove the preservation all such formulas of linear temporal logics not containg the operator `next` ($\bigcirc$)
- `PROD` supports the stubborn set method with different algorithmic solutions.
- The most practical and efficient way of utilising stubborn sets is to use testers and execute the reachability graph generator with the option `-C`.

©Johan Lilius, Keijo Heljanko, Esa Kettunen 1997–2000