

# ASR : an introduction to Distributed Systems and Algorithms (Algorithmes et Systèmes Répartis) 2/3

M. Raynal<sup>1</sup>, A. Mostefaoui<sup>2</sup>, E. Fabre<sup>3</sup>

<sup>1</sup>Pr., ASAP team

<sup>2</sup>MC, ASAP team

<sup>3</sup>CR INRIA, DistribCom team

Master 2, research in Computer Science, '07

Download the **lecture slides** from my web-page :

[http://www.irisa.fr/distribcom/Personal\\_Pages/fabre/fabre.html](http://www.irisa.fr/distribcom/Personal_Pages/fabre/fabre.html)







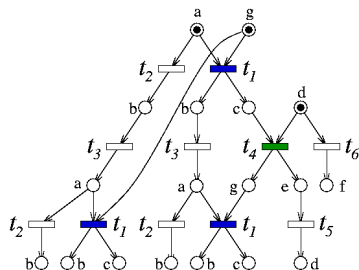












## Concurrency:

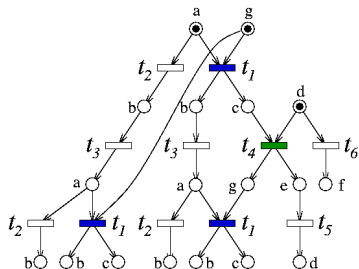
$$x \perp\!\!\!\perp y \Leftrightarrow \neg(x \rightarrow^* y) \wedge \neg(y \rightarrow^* x) \wedge \neg(x \# y)$$

**Co-set:**  $X \subseteq C$  such that  $\forall c, c' \in X, c \perp\!\!\!\perp c'$

**Cut:** a maximal co-set for  $\subseteq$

**Prefix:**  $O' = (C', E', \dots) \sqsubseteq O$  iff

$O'$  is a causally-closed sub-net of  $O$ , containing  $C_0$  and  $E'^*$



## Concurrency:

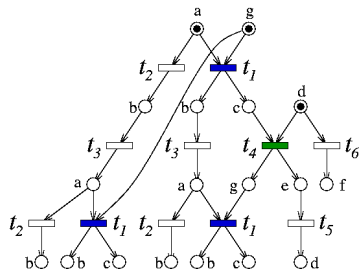
$$x \perp\!\!\!\perp y \Leftrightarrow \neg(x \rightarrow^* y) \wedge \neg(y \rightarrow^* x) \wedge \neg(x \# y)$$

**Co-set:**  $X \subseteq C$  such that  $\forall c, c' \in X, c \perp\!\!\!\perp c'$

**Cut:** a maximal co-set for  $\subseteq$

**Prefix:**  $O' = (C', E', \dots) \sqsubseteq O$  iff

$O'$  is a causally-closed sub-net of  $O$ , containing  $C_0$  and  $E'^*$



## Concurrency:

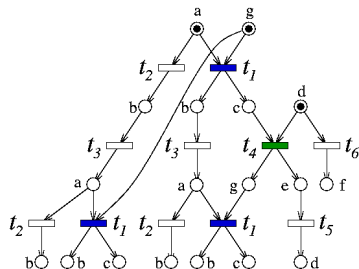
$$x \perp\!\!\!\perp y \Leftrightarrow \neg(x \rightarrow^* y) \wedge \neg(y \rightarrow^* x) \wedge \neg(x \# y)$$

**Co-set:**  $X \subseteq C$  such that  $\forall c, c' \in X, c \perp\!\!\!\perp c'$

**Cut:** a maximal co-set for  $\subseteq$

**Prefix:**  $O' = (C', E', \dots) \sqsubseteq O$  iff

$O'$  is a causally-closed sub-net of  $O$ , containing  $C_0$  and  $E'^*$



## Concurrency:

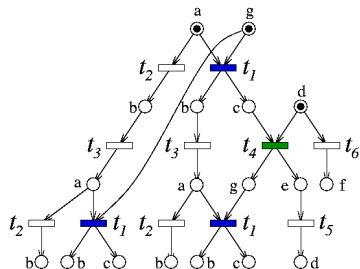
$$x \perp\!\!\!\perp y \Leftrightarrow \neg(x \rightarrow^* y) \wedge \neg(y \rightarrow^* x) \wedge \neg(x \# y)$$

**Co-set:**  $X \subseteq C$  such that  $\forall c, c' \in X, c \perp\!\!\!\perp c'$

**Cut:** a maximal co-set for  $\subseteq$

**Prefix:**  $O' = (C', E', \dots) \sqsubseteq O$  iff

$O'$  is a causally-closed sub-net of  $O$ , containing  $C_0$  and  $E'^*$



## Concurrency:

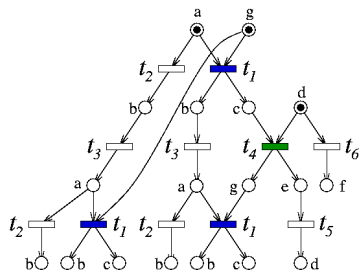
$$x \perp\!\!\!\perp y \Leftrightarrow \neg(x \rightarrow^* y) \wedge \neg(y \rightarrow^* x) \wedge \neg(x \# y)$$

**Co-set:**  $X \subseteq C$  such that  $\forall c, c' \in X, c \perp\!\!\!\perp c'$

**Cut:** a maximal co-set for  $\subseteq$

**Prefix:**  $O' = (C', E', \dots) \sqsubseteq O$  iff

$O'$  is a causally-closed sub-net of  $O$ , containing  $C_0$  and  $E'^\bullet$

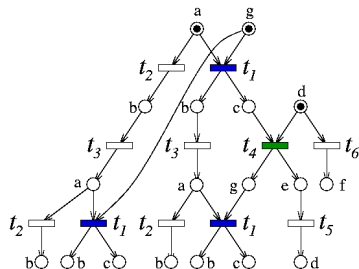


**Configuration:** denoted  $\kappa$ , a conflict-free prefix of  $\mathcal{O}$

- specific case: local configuration  
 $[e]$  = smallest configuration containing event  $e$

Lemma (cuts and configurations)

$X$  is a cut of  $\mathcal{O}$   $\Leftrightarrow \exists \kappa = (C', E', \dots)$  such that  $X = \max(C')$ .

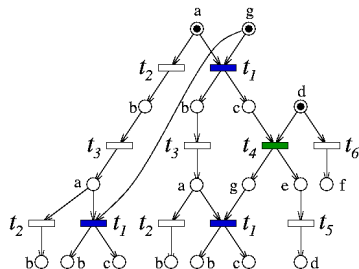


**Configuration:** denoted  $\kappa$ , a conflict-free prefix of  $\mathcal{O}$

- specific case: local configuration  
 $[e]$  = smallest configuration containing event  $e$

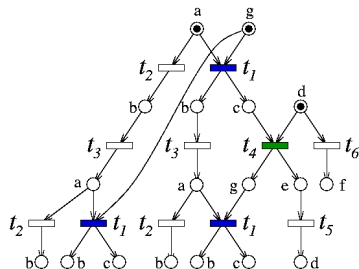
**Lemma (cuts and configurations)**

$X$  is a cut of  $\mathcal{O}$   $\Leftrightarrow \exists \kappa = (C', E', \dots)$  such that  $X = \max(C')$ .



**Branching process of  $\mathcal{N}$ :**  $(\mathcal{O}, f)$  such that

- $f : \mathcal{O} \rightarrow \mathcal{N}$  is a morphism of Petri nets, and a total function,
- $f$  labels conditions / events of  $\mathcal{O}$  by places / transitions of  $\mathcal{N}$
- **parsimony:**  $\forall e, e' \in E, \bullet e = \bullet e', f(e) = f(e') \Rightarrow e = e'$
- Through  $f$ , configurations of  $\mathcal{O}$  represent runs of  $\mathcal{N}$
- Let  $X =$  maximal conditions of  $\kappa$  ( $X$  forms a cut),  
 $f(X)$  is the marking of  $\mathcal{N}$  produced by run  $\kappa$ .



**Branching process of  $\mathcal{N}$ :**  $(\mathcal{O}, f)$  such that

- $f : \mathcal{O} \rightarrow \mathcal{N}$  is a morphism of Petri nets, and a total function,
- $f$  labels conditions / events of  $\mathcal{O}$  by places / transitions of  $\mathcal{N}$
- **parsimony:**  $\forall e, e' \in E, \bullet e = \bullet e', f(e) = f(e') \Rightarrow e = e'$
- Through  $f$ , configurations of  $\mathcal{O}$  represent runs of  $\mathcal{N}$
- Let  $X =$  maximal conditions of  $\kappa$  ( $X$  forms a cut),  
 $f(X)$  is the marking of  $\mathcal{N}$  produced by run  $\kappa$ .

## Theorem

*There exists a unique maximal branching process  $(\mathcal{U}_N, f_N)$  of  $N$ , maximal for the prefix relation  $\sqsubseteq$ . It is called the **unfolding** of  $N$ .*

Proof: by definition of the union of BP of a net  $N$ .

### Unfolding algorithm:

- Init:
  - $C = C_0$  isomorphic to  $P_0$  by  $f_N$
  - $E = \emptyset, \rightarrow = \emptyset$
- Until stability, repeat
  - for a co-set  $X \subseteq C$  and transition  $t \in T$  such that  $f_N(X) = \bullet t$
  - create event  $e$  in  $E$  (if it doesn't already exist) with  $f_N(e) = t, \bullet(e) = X$
  - then add new conditions  $X'$  to  $C$  with  $X' = e^\bullet$ , and  $X'$  isomorphic to  $t^\bullet$  by  $f_N$

## Theorem

*There exists a unique maximal branching process  $(\mathcal{U}_N, f_N)$  of  $N$ , maximal for the prefix relation  $\sqsubseteq$ . It is called the **unfolding** of  $N$ .*

Proof: by definition of the union of BP of a net  $N$ .

## Unfolding algorithm:

- Init:
  - $C = C_0$  isomorphic to  $P_0$  by  $f_N$
  - $E = \emptyset, \rightarrow = \emptyset$
- Until stability, repeat
  - for a co-set  $X \subseteq C$  and transition  $t \in T$  such that  $f_N(X) = \bullet t$
  - create event  $e$  in  $E$  (if it doesn't already exist)
    - with  $f_N(e) = t, \bullet(e) = X$
  - then add new conditions  $X'$  to  $C$ 
    - with  $X' = e^\bullet$ , and  $X'$  isomorphic to  $t^\bullet$  by  $f_N$

# Finite complete prefix

**Objective:** select a finite prefix of  $\mathcal{U}_N$  that is sufficient to test properties of the concurrent system (or distributed application).

**Idea:** avoid repeating the same patterns in the unfolding.

## Definition (FCP)

A finite prefix  $O$  of  $\mathcal{U}_N$  is complete iff

- all reachable markings are represented

$$\forall M \text{ reachable in } \mathcal{N}, \exists \kappa \in O, M = \text{Mark}(\kappa) = f_N[\max(\kappa)]$$

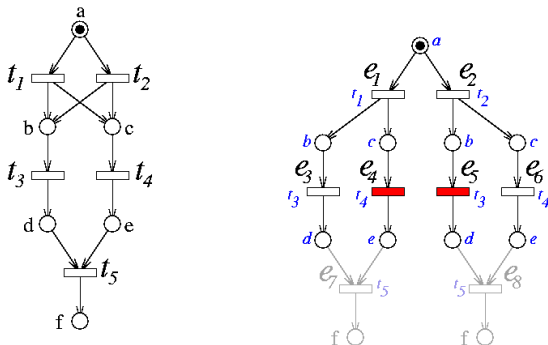
- all transitions firable from a reachable marking appear

$$\forall t \in T, M[t\rangle M', \exists \kappa, \kappa' \in O, M = \text{Mark}(\kappa), \kappa' = \kappa \oplus \{e\} \text{ and } f_N(e) = t$$

**Naive idea:** apply the unfolding algorithm, and stop it as soon as an already met marking is found.

- Stop criterion:  $Mark([e])$  exists in  $O$ ,
- this makes  $e$  a **cut-off event**

**Problem:** generally yields an incomplete prefix...



**Solution:** break the symmetry, by favoring some configurations

**Adequate order:**  $<$  on (local) configurations

- well founded (finite number of predecessors for any configuration)
- refines prefix inclusion:  $\kappa \sqsubset \kappa' \Rightarrow \kappa < \kappa'$
- preserved by isomorphic extension:  
if  $\kappa < \kappa'$  and  $Mark(\kappa) = Mark(\kappa')$   
then  $\kappa \oplus e < \kappa' \oplus e'$  where  $f_N(e) = f_N(e')$

### Examples

- take  $< = \sqsubset$
- the number of events (Mc Millan)
- the lexicographic order, for product nets (Esparza):
  - events of each component are counted separately
  - to each configuration is attached the vector of countings
  - the lexicographic order is based on an ordering of component

**Solution:** break the symmetry, by favoring some configurations

**Adequate order:**  $<$  on (local) configurations

- well founded (finite number of predecessors for any configuration)
- refines prefix inclusion:  $\kappa \sqsubset \kappa' \Rightarrow \kappa < \kappa'$
- preserved by isomorphic extension:  
if  $\kappa < \kappa'$  and  $Mark(\kappa) = Mark(\kappa')$   
then  $\kappa \oplus e < \kappa' \oplus e'$  where  $f_N(e) = f_N(e')$

## Examples

- take  $< = \sqsubset$
- the number of events (Mc Millan)
- the lexicographic order, for product nets (Esparza):
  - events of each component are counted separately
  - to each configuration is attached the vector of countings
  - the lexicographic order is based on an ordering of component

## Definition (Cut-off event)

*Event  $e$  is a cut-off event in  $BP(O, f_N)$  of  $N$  iff there exists another  $e'$  in  $O$  such that  $Mark([e']) = Mark([e])$  and  $[e'] < [e]$ .*

## Prefix construction:

- Init:
  - $C = C_0$  isomorphic to  $P_0$  by  $f_N$
  - $E = \emptyset, \rightarrow = \emptyset$
- Until stability, repeat
  - for a co-set  $X \subseteq C$  and transition  $t \in T$  such that  $f_N(X) = \bullet t$   
and such that there is no cut-off event in  $[X]$
  - create event  $e$  in  $E$  (if it doesn't already exist)  
with  $f_N(e) = t, \bullet(e) = X$
  - then add new conditions  $X'$  to  $C$   
with  $X' = e^\bullet$ , and  $X'$  isomorphic to  $t^\bullet$  by  $f_N$

## Definition (Cut-off event)

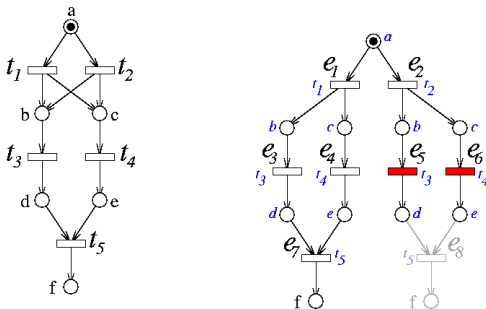
*Event  $e$  is a cut-off event in  $BP(O, f_N)$  of  $N$  iff there exists another  $e'$  in  $O$  such that  $Mark([e']) = Mark([e])$  and  $[e'] < [e]$ .*

## Prefix construction:

- Init:
  - $C = C_0$  isomorphic to  $P_0$  by  $f_N$
  - $E = \emptyset, \rightarrow = \emptyset$
- Until stability, repeat
  - for a co-set  $X \subseteq C$  and transition  $t \in T$  such that  $f_N(X) = \bullet t$   
and such that there is no cut-off event in  $[X]$
  - create event  $e$  in  $E$  (if it doesn't already exist)  
with  $f_N(e) = t, \bullet(e) = X$
  - then add new conditions  $X'$  to  $C$   
with  $X' = e^\bullet$ , and  $X'$  isomorphic to  $t^\bullet$  by  $f_N$

## Application: resolution of the previous counter-example

- Assume  $[e_3] < [e_5]$ , which makes  $e_5$  a cut-off so  $[e_3, e_4] < [e_5, e_6]$  (isomorphic extensions).
- $[e_6] < [e_4]$  is false: it would induce  $[e_6, e_5] < [e_4, e_3]$ ! So  $e_4$  can't be a cut-off event.
- The algorithm reaches  $e_7$ .
- Notice that either  $[e_4] < [e_6]$ , which makes  $e_6$  a cut-off, or  $[e_4]$  and  $[e_6]$  are incomparable.



## Theorem

*The prefix  $O$  obtained by stopping the unfolding algorithm at cut-off events is finite and complete.*

### Proof of completeness.

- $M$  reachable in  $\mathcal{N} \Rightarrow \exists \kappa \sqsubseteq \mathcal{U}_{\mathcal{N}}, \text{Mark}(\kappa) = M$
- if  $\kappa$  contains no cut-off, then  $\kappa \sqsubseteq O$
- otherwise let  $e \in \kappa$  be a cut-off, with **corresponding event** of  $e_1$   
 $[e_1] < [e]$  and  $\text{Mark}([e_1]) = \text{Mark}([e])$   
 consider  $\kappa_1 = [e_1] \oplus (\kappa \setminus [e])$ , then  $\kappa_1 < \kappa$  and  $\text{Mark}(\kappa_1) = M$   
 this removes  $e$  from  $\kappa$
- If  $\kappa_1$  contains a cut-off, repeat the construction to build a  $\kappa_2$ .
- The process necessarily stops, because  $\kappa > \kappa_1 > \kappa_2 > \dots$   
 and  $<$  is well-founded.

## Theorem

*The prefix  $\mathcal{O}$  obtained by stopping the unfolding algorithm at cut-off events is finite and complete.*

### Proof of finiteness.

- $\mathcal{O}$  has finite “depth” (= maximum length of a causal sequence  $e_1 \rightarrow^* e_2 \rightarrow^* \dots \rightarrow^* e_n$ )  
there is a finite number of reachable markings in  $\mathcal{N}$   
two consecutive events reaching the same marking are comparable for  $<$
- $\mathcal{O}$  has a finite “width”  
 $C_0$  is finite,  $\bullet e$  and  $e \bullet$  are finite,  
so there is a finite number of events that have depth lower than  $k$  (proof by recursion)

## Size of a finite complete prefix:

Is it more interesting to use a FCP or a marking graph to check the accessibility of some marking  $M$  of  $\mathcal{N}$  ?

### Proposition

*If the adequate order used to build the FCP  $O$  is a total order, then the number of non cut-off events in  $O$  is bounded by the number of reachable markings.*

**Proof.** For  $e, e' \in O$ , if  $Mark([e]) = Mark([e'])$ , given that either  $[e] < [e']$  or  $[e'] < [e]$ , one of them is a cut-off.

Since cut-off events can only be on top of a few non cut-off events, this also bounds the size of the FCP.

## More on finite complete prefixes

- The algorithmic construction may yield different FCP...  
There exists a **canonical construction**,  
that defines cut-offs “off-line” on  $\mathcal{U}_N$  (Khomenko).
- A FCP of  $\mathcal{U}_N$  is sufficient to recover the full unfolding  $\mathcal{U}_N$ .  
Notion of **e-shift**
  - let  $e$  be a cut-off, with corresponding event  $e'$ ,
  - let  $\uparrow [e'] = \{x \in \mathcal{O} : [e'] \rightarrow^* x\}$  be the “future” of  $[e']$
  - paste a copy of  $\uparrow [e']$  after  $[e]$

Repeating this e-shift for all cut-offs rebuilds the complete unfolding.

# Model-checking applications, and complexity

## 1. Reachability tests

Is the (sub-)marking  $M$  reachable in net  $\mathcal{N}$  ?

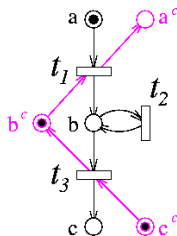
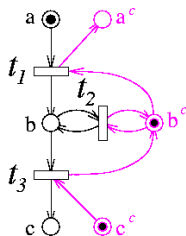
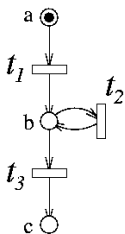
By completeness, it suffices to check this on a FCP of  $\mathcal{N}$ .

### Sub-marking:

Try to reach  $(Q_1, Q_0)$ , i.e. a marking where places in  $Q_1$  are marked, and places in  $Q_0$  are empty.

**Remark 1:** By introducing **complementary places**  $p^c$  to  $\mathcal{N}$ ,  $\forall p \in P$ , becomes equivalent to assuming  $Q_0 = \emptyset$ .

- exactly one of  $\{p, p^c\}$  is marked at each time
- $t \in \bullet p \Leftrightarrow t \in p^c \bullet$  and  $t \in p \bullet \Leftrightarrow t \in \bullet p^c$
- excepted for  $t \in \bullet p \cap p \bullet$



**Remark 2:**

- Looking for the submarking  $(Q_1, \emptyset)$  amounts to checking that some extra transition  $t^f$  with  $\bullet t^f = Q_1$  is firable in  $\mathcal{N}$ .
- This is equivalent to looking for a co-set on which  $t^f$  can be fired in the unfolding algorithm.
- So reachability and co-set construction have identical complexities.

## Proposition

*The reachability test (co-set construction) is NP-hard.*

**Remark 2:**

- Looking for the submarking  $(Q_1, \emptyset)$  amounts to checking that some extra transition  $t^f$  with  $\bullet t^f = Q_1$  is firable in  $\mathcal{N}$ .
- This is equivalent to looking for a co-set on which  $t^f$  can be fired in the unfolding algorithm.
- So reachability and co-set construction have identical complexities.

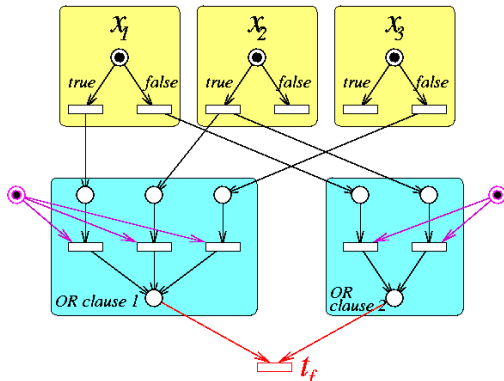
**Proposition**

*The reachability test (co-set construction) is NP-hard.*

**Proof:** it is equivalent to solving a SAT problem.

Example:

encoding of the SAT problem  $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2)$   
in a safe Petri net



The complexity of moving to the unfolding is polynomial.  
So finding a co-set where  $t_f$  is firable is NP-hard.

**Algorithm:** to build a co-set  $X$  with  $f_N(X) = Q_1 = \{p_1, p_2, \dots\}$

- Method

- take a condition  $c_1$  with  $f_N(c_1) = p_1$
- extend it with a condition  $c_2$  with  $f_N(c_2) = p_2$ , and  $c_2 \perp\!\!\!\perp c_1$
- try to add  $c_3$  with  $f_N(c_3) = p_3$ , and  $c_3 \perp\!\!\!\perp \{c_1, c_2\}$
- etc., backtracking when failing

- Concurrency test for  $c_1 \perp\!\!\!\perp c_2$

- build  $[c_1]$  and  $[c_2]$
- check  $[c_1] \cap [c_2] \cap C = \emptyset$  i.e. no conflict
- check  $\neq (c_1 \in [c_2])$ , i.e. not  $c_1 \rightarrow^* c_2$
- check  $\neq (c_2 \in [c_1])$ , i.e. not  $c_2 \rightarrow^* c_1$

- The best implementations use SAT solvers for these properties.

- These ideas also used to build unfoldings/prefixes. Modern implementations use and update a list of possible extensions (= event + co-set).

**Algorithm:** to build a co-set  $X$  with  $f_N(X) = Q_1 = \{p_1, p_2, \dots\}$

- Method
  - take a condition  $c_1$  with  $f_N(c_1) = p_1$
  - extend it with a condition  $c_2$  with  $f_N(c_2) = p_2$ , and  $c_2 \perp\!\!\!\perp c_1$
  - try to add  $c_3$  with  $f_N(c_3) = p_3$ , and  $c_3 \perp\!\!\!\perp \{c_1, c_2\}$
  - etc., backtracking when failing
- Concurrency test for  $c_1 \perp\!\!\!\perp c_2$ 
  - build  $[c_1]$  and  $[c_2]$
  - check  $[c_1] \cap [c_2] \cap C = \emptyset$  i.e. no conflict
  - check  $\neq (c_1 \in [c_2])$ , i.e. not  $c_1 \rightarrow^* c_2$
  - check  $\neq (c_2 \in [c_1])$ , i.e. not  $c_2 \rightarrow^* c_1$
- The best implementations use SAT solvers for these properties.
- These ideas also used to build unfoldings/prefixes. Modern implementations use and update a list of possible extensions (= event + co-set).

**Another approach:** using the **marking equation**

- let  $\omega = t^1, \dots, t^n$  be a firable sequence in net  $\mathcal{N}$
- the **Parikh vector**  $\vec{\omega} = [\nu(\omega, t)]_{t \in T}$  counts occurrences of each transition  $t$  of  $T$  in  $\omega$
- the **incidence matrix**  $N \in \mathbb{N}^{|P| \times |T|}$  is defined by

$$N(p, t) = \mathbf{1}_{t \rightarrow p} - \mathbf{1}_{p \rightarrow t}$$

- the marking  $M$  reached by  $\omega$  is given by

$$M = P_0 + N \cdot \vec{\omega}$$

**Proposition**

*$M$  is reachable only iff there exists a solution to  $M = P_0 + N \cdot X$ ,  $X \in \mathbb{N}^{|T|}$ . If the net  $\mathcal{N}$  is acyclic, this condition is also sufficient.*

Can be used on a finite complete prefix, assuming it is already available. Resolution by an **integer linear program solver**.

**Another approach:** using the **marking equation**

- let  $\omega = t^1, \dots, t^n$  be a firable sequence in net  $\mathcal{N}$
- the **Parikh vector**  $\vec{\omega} = [\nu(\omega, t)]_{t \in T}$  counts occurrences of each transition  $t$  of  $T$  in  $\omega$
- the **incidence matrix**  $N \in \mathbb{N}^{|P| \times |T|}$  is defined by

$$N(p, t) = \mathbf{1}_{t \rightarrow p} - \mathbf{1}_{p \rightarrow t}$$

- the marking  $M$  reached by  $\omega$  is given by

$$M = P_0 + N \cdot \vec{\omega}$$

**Proposition**

*$M$  is reachable only iff there exists a solution to  $M = P_0 + N \cdot X$ ,  $X \in \mathbb{N}^{|T|}$ . If the net  $\mathcal{N}$  is acyclic, this condition is also sufficient.*

Can be used on a finite complete prefix, assuming it is already available. Resolution by an **integer linear program solver**.

## 2. Deadlock checking:

**Deadlock** = marking of  $\mathcal{N}$  where no more transition can be fired.

### Proposition (Mc Millan)

*Let  $O$  be a FCP of  $\mathcal{U}_{\mathcal{N}}$ . There exists no deadlock in  $\mathcal{N}$  iff every configuration  $\kappa$  of  $O$  can be extended into a configuration  $\kappa' \supseteq \kappa$  that contains a cut-off event.*

This is simply because there is no deadlock iff every configuration can be made arbitrarily large in  $\mathcal{U}_{\mathcal{N}}$  (recall the notion of e-shift).

### Corollary

*Equivalently, there is a deadlock in  $\mathcal{N}$  iff there exists some  $\kappa$  that can't be extended to reach a cut-off.*

## 2. Deadlock checking:

**Deadlock** = marking of  $\mathcal{N}$  where no more transition can be fired.

### Proposition (Mc Millan)

*Let  $O$  be a FCP of  $\mathcal{U}_{\mathcal{N}}$ . There exists no deadlock in  $\mathcal{N}$  iff every configuration  $\kappa$  of  $O$  can be extended into a configuration  $\kappa' \supseteq \kappa$  that contains a cut-off event.*

This is simply because there is no deadlock iff every configuration can be made arbitrarily large in  $\mathcal{U}_{\mathcal{N}}$  (recall the notion of e-shift).

### Corollary

*Equivalently, there is a deadlock in  $\mathcal{N}$  iff there exists some  $\kappa$  that can't be extended to reach a cut-off.*

## Corollary

*Equivalently, there is a deadlock in  $\mathcal{N}$  iff there exists some  $\kappa$  that can't be extended to reach a cut-off.*

### Proof of $\Leftarrow$

- extend  $\kappa$  into a maximal config.  $\kappa'$ , that contains no cut-off
- no more transition can be fired after  $\kappa'$  (otherwise one of its maximal events would be a cut-off)
- so  $Mark(\kappa')$  is a deadlock

### Proof of $\Rightarrow$

- let  $M$  be a deadlock and  $\kappa \in \mathcal{O}$  such that  $Mark(\kappa) = M$
- there exists one such  $\kappa$  that contains no cut-off (remove cut-offs one by one, as in the completeness proof of a FCP)
- this  $\kappa$  can't be extended at all (otherwise  $M$  wouldn't be a deadlock)

## Corollary

*Equivalently, there is a deadlock in  $\mathcal{N}$  iff there exists some  $\kappa$  that can't be extended to reach a cut-off.*

### Proof of $\Leftarrow$

- extend  $\kappa$  into a maximal config.  $\kappa'$ , that contains no cut-off
- no more transition can be fired after  $\kappa'$  (otherwise one of its maximal events would be a cut-off)
- so  $Mark(\kappa')$  is a deadlock

### Proof of $\Rightarrow$

- let  $M$  be a deadlock and  $\kappa \in \mathcal{O}$  such that  $Mark(\kappa) = M$
- there exists one such  $\kappa$  that contains no cut-off (remove cut-offs one by one, as in the completeness proof of a FCP)
- this  $\kappa$  can't be extended at all (otherwise  $M$  wouldn't be a deadlock)

## Corollary

*Let  $\kappa$  be a configuration in conflict with all cut-off events of  $\mathcal{O}$ .  
Let  $\kappa' \sqsupseteq \kappa$  be a maximal extension of  $\kappa$  in  $\mathcal{O}$ . Then  $\text{Mark}(\kappa')$  is a deadlock.*

Homework: does this characterize *all* deadlocks ?

- checking the existence of deadlocks is also NP hard
- there exist “graphical” methods on the FCP  $\mathcal{O}$
- there exist as well methods based on the marking equation



# Bibliography

## About Prefix constructions

- An unfolding algorithm for synchronous products of transition systems, Esparza and Romer, proceedings of CONCUR'99, pp 2-20
- An improvement of McMillan's unfolding algorithm, Esparza and Romer, LNCS 1055, pp 87-106, 1996
- Canonical prefixes of Petri net unfoldings, Khomenko, Koutny and Vogler, Acta Informatica 40, pp 95-118, 2003

## More oriented to model-checking applications

- Model checking using net unfoldings, Esparza, Science of Computer Programming 23, pp 151-195, 1994
- Reachability analysis using net unfoldings, Schroter and Esparza,
- Deadlock checking using net unfoldings, Melzer and Romer, LNCS 1254, pp 352-363, 1997
- Using net unfoldings to avoid the state explosion problem in the verification of asynchronous circuits, McMillan, LNCS 663, pp 164-174, 1992