

# Petri Nets (for Planners)

ICAPS 2009 —  
Introduction

## 1-Safe Petri Nets: Basic Definitions

Transition Systems

Petri Nets

Petri Nets for Planning

## Unfolding

Transition Systems

Products and Petri Nets

Branching Processes

Verification

Construction

Search Procedures

Search in Transition Systems

Search in Product Systems

Directed Unfolding

## Planning via Unfolding & Concurrency

(... from various places ...)  
From Planning Problem to Petri Net

Concurrency, Plan Flexibility & Makespan

## General Petri Nets

ICAPS 2009 1 / 140

# Petri Nets (for Planners)

B. Bonet, P. Haslum, S. Hickmott,  
S. Thiébaux, S. Edelkamp

(... from various places ...)

ICAPS 2009

(... from various places ...)

ICAPS 2009 2 / 140

Introduction

# Introduction & Motivation

- ▶ Petri Nets (PNs) is a formalism for modelling discrete event systems.
  - As are planning formalisms (STRIPS, SAS+, etc).
  - Important differences: general Petri nets are infinite, different models of event concurrency.
- ▶ Developed by (and named after) C.A. Petri in 1960s.
- ▶ An exchange of ideas between Petri net theory and planning holds potential to benefit both:
  - A wealth of results (theoretical and practical) exist for Petri nets.
  - Yet, some standard planning techniques (e.g., search heuristics) are unheard of in the PN community.

(... from various places ...)

ICAPS 2009 3 / 140

Introduction

# Outline of the Tutorial

1. 1-Safe Petri Nets.
  - 1.1 1-Safe nets as a representation of products of transition systems.
2. Unfolding: An Analysis Method for 1-Safe Nets.
  - 2.1 Unfoldings and branching processes.
  - 2.2 Constructing the unfolding: search.
  - 2.3 Planning via unfolding.
  - 2.4 Concurrency properties of the generated plans.
3. General Petri Nets.
  - 3.1 Modelling and expressivity.
  - 3.2 Analysis methods for general Petri nets.
  - 3.3 Petri nets with special structure.
4. Conclusions

(... from various places ...)

ICAPS 2009 4 / 140

## Part 1: Introduction to 1-Safe Petri Nets

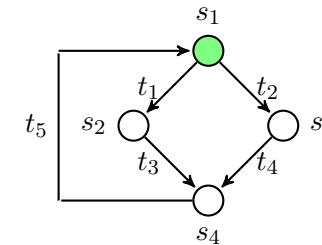
- ▶ 1-safe Petri nets is a class of Petri nets that is closely related to planning formalisms.
  - Compact representation of products of sequential transition systems.

(... from various places ...)

ICAPS 2009 5 / 140

## Transition systems 1/2

- ▶ Transition systems used to model sequential systems

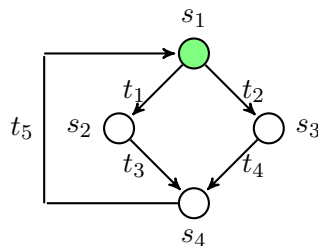


- ▶ A tuple  $\mathcal{A} = \langle S, T, \alpha, \beta, is \rangle$  where  $S$  and  $T$  are states and transitions,  $\alpha$  and  $\beta$  are source and target states, and  $is$  is the initial state
- ▶ E.g.,  $\alpha(t_4) = s_3$ ,  $\beta(t_1) = s_2$ , and  $is = s_1$

(... from various places ...)

ICAPS 2009 6 / 140

## Transition systems 2/2



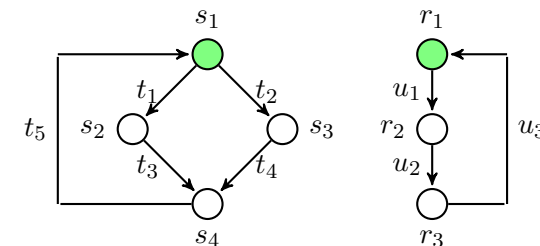
- ▶ The triplet  $\langle \alpha(t), t, \beta(t) \rangle$  is a **step**; e.g.  $\langle s_2, t_3, s_4 \rangle$
- ▶ A “transition word”  $t_1 t_2 \dots t_k$  is a **computation** if there is sequence  $s_0 s_1 \dots s_k$  so that  $\langle s_i, t_i, s_{i+1} \rangle$  is a step
- ▶ A computation is a **history** if  $s_0 = is$
- ▶ Computation and histories may be infinite; e.g.  $t_1 t_3 t_5 t_1 t_3 t_5 \dots$  is an infinite history

(... from various places ...)

ICAPS 2009 7 / 140

## (Synchronised) products of transition systems 1/2

- ▶ Model concurrent systems with multiple components



- ▶ Let  $\mathcal{A}_1, \dots, \mathcal{A}_n$  be transition systems. A **synchronisation constraint**  $\mathbf{T}$  is a subset of

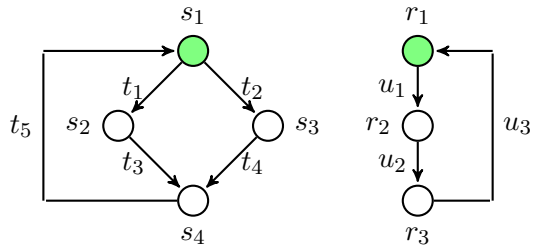
$$(T_1 \cup \{\epsilon\}) \times \dots \times (T_n \cup \{\epsilon\}) \setminus \{\langle \epsilon, \dots, \epsilon \rangle\}$$

- ▶ Each  $t \in T$  is a global transition
- ▶ If  $t_i \neq \epsilon$ ,  $\mathcal{A}_i$  **participates** in  $t$
- ▶ The initial global state is equals  $\langle is_1, \dots, is_n \rangle$

(... from various places ...)

ICAPS 2009 8 / 140

### (Synchronised) products of transition systems 2/2



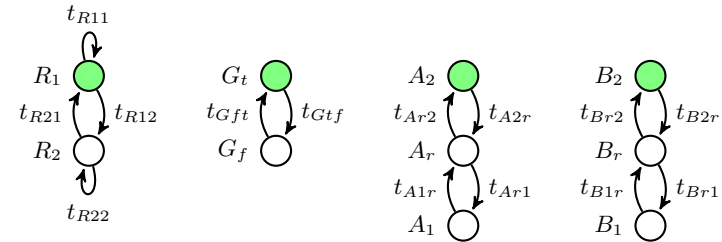
- ▶  $\mathbf{T} = \{ \langle t_1, \epsilon \rangle, \langle t_2, \epsilon \rangle, \langle t_3, u_2 \rangle, \langle t_4, u_2 \rangle, \langle t_5, \epsilon \rangle, \langle \epsilon, u_1 \rangle, \langle \epsilon, u_3 \rangle \}$  is a synchronisation constraint
- ▶ (Global) steps, computations and histories are defined like before; e.g.  $\langle t_1, \epsilon \rangle \langle \epsilon, u_1 \rangle \langle t_3, u_2 \rangle$  is a computation and history

(... from various places ...)

### Transition systems for Gripper with one arm 1/2

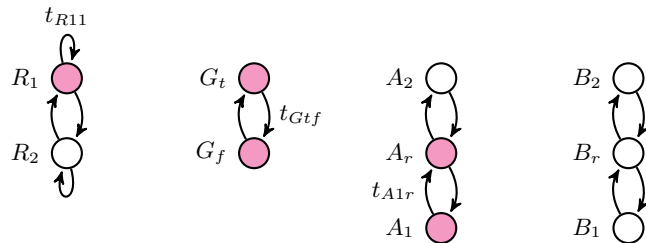
**Variables:**

- ▶ Position of Robot:  $R_1, R_2$
- ▶ Empty gripper:  $G_t, G_f$
- ▶ Position of ball A:  $A_1, A_2, A_r$
- ▶ Position of ball B:  $B_1, B_2, B_r$



(... from various places ...)

### Transition systems for Gripper with one arm 2/2

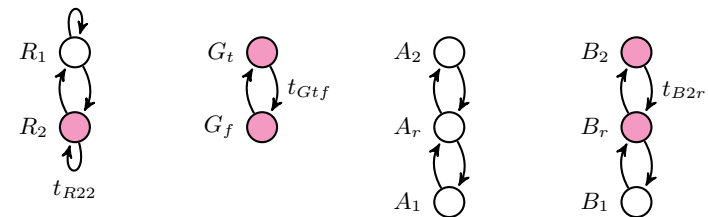


**Synchronisation Constraints:**

$$\text{pickup}(A, 1) = \langle t_{R11}, t_{Gtf}, t_{A1r}, \epsilon \rangle$$

(... from various places ...)

### Transition systems for Gripper with one arm 2/2



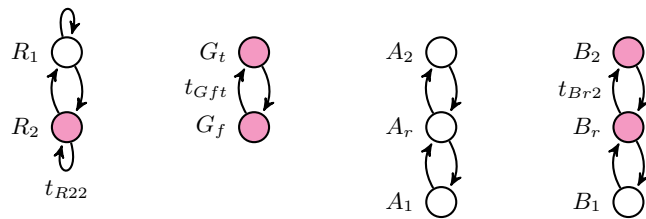
**Synchronisation Constraints:**

$$\text{pickup}(A, 1) = \langle t_{R11}, t_{Gtf}, t_{A1r}, \epsilon \rangle$$

$$\text{pickup}(B, 2) = \langle t_{R22}, t_{Gtf}, \epsilon, t_{B2r} \rangle$$

(... from various places ...)

## Transition systems for Gripper with one arm 2/2

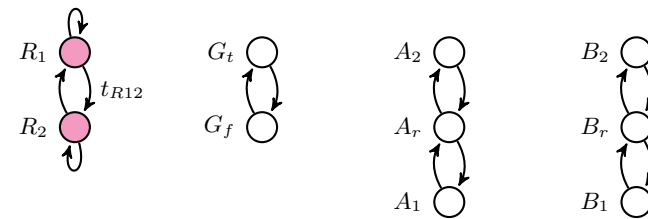


### Synchronisation Constraints:

$$\begin{aligned} \text{pickup}(A, 1) &= \langle t_{R11}, t_{Gtf}, t_{A1r}, \epsilon \rangle \\ \text{pickup}(B, 2) &= \langle t_{R22}, t_{Gtf}, \epsilon, t_{B2r} \rangle \\ \text{drop}(B, 2) &= \langle t_{R22}, t_{Gft}, \epsilon, t_{Br2} \rangle \end{aligned}$$

(... from various places ...)

## Transition systems for Gripper with one arm 2/2



### Synchronisation Constraints:

$$\begin{aligned} \text{pickup}(A, 1) &= \langle t_{R11}, t_{Gtf}, t_{A1r}, \epsilon \rangle \\ \text{pickup}(B, 2) &= \langle t_{R22}, t_{Gtf}, \epsilon, t_{B2r} \rangle \\ \text{drop}(B, 2) &= \langle t_{R22}, t_{Gft}, \epsilon, t_{Br2} \rangle \\ \text{move}(1, 2) &= \langle t_{R12}, \epsilon, \epsilon, \epsilon \rangle \end{aligned}$$

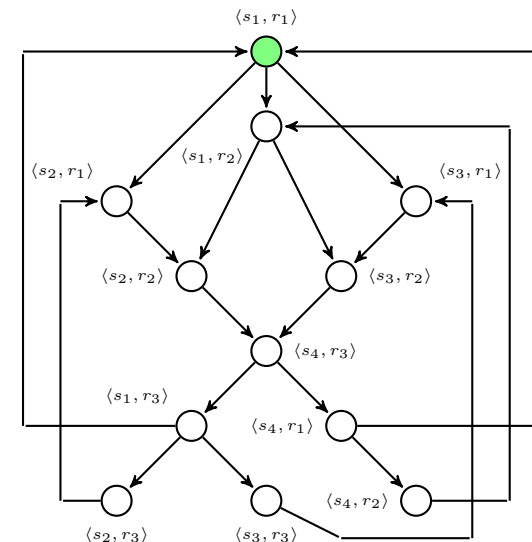
(... from various places ...)

## Semantics for products: Interleaving semantics

- ▶ A product  $\mathbf{A} = \langle \mathcal{A}_1, \dots, \mathcal{A}_n, \mathbf{T} \rangle$  can be translated into an equivalent transition system  $\mathcal{T}_{\mathbf{A}} = \langle S, T, \alpha, \beta, is \rangle$  where
  - $S$  is the set of global states of  $\mathbf{A}$
  - $T$  is the set of steps  $\langle s, t, s' \rangle$
  - $\alpha(\langle s, t, s' \rangle) = s$  and  $\beta(\langle s, t, s' \rangle) = s'$
  - $is = \mathbf{is}$
- ▶ The interleaving semantics is of exponential size

(... from various places ...)

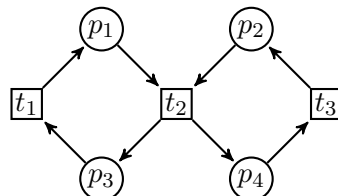
## Interleaving semantics: Example



(... from various places ...)

### Petri nets 1/5

- ▶ A Petri net is a bipartite graph, with nodes divided into *places* (circles) and *transitions* (boxes)



- ▶ Formally, a tuple  $N = \langle P, T, F \rangle$  where  $P / T$  are the sets of places / transitions and  $F \subseteq (P \times T) \cup (T \times P)$  is the *flow* (i.e., edge) *relation*
- ▶ For any node  $n \in P \cup T$ ,  $\bullet n = \{n' \mid (n', n) \in F\}$  and  $n^\bullet = \{n' \mid (n, n') \in F\}$  are the inputs and outputs of  $n$

(... from various places ...)

### Petri nets 2/5

- ▶ The state of a Petri net  $N = \langle P, T, F \rangle$  is defined by a *marking*, which puts zero or more *tokens* on each place. Formally, a marking is a mapping  $\mathbf{m} : P \rightarrow \mathbb{N}$
- ▶ Transition  $t$  is *enabled* at marking  $\mathbf{m}$  iff  $\mathbf{m}(p) > 0$  for each  $p \in \bullet t$ , i.e., iff every input of  $t$  is marked
  - Notation:  $\mathbf{m} [t]$
- ▶ If  $t$  is enabled it can *fire* (or *occur*), leading to a new marking  $\mathbf{m}'$  such that
 

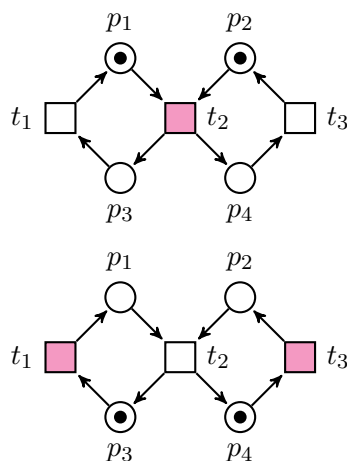
$\mathbf{m}'(p) = \mathbf{m}(p) - 1$	if $p \in \bullet t$ (and $p \notin t^\bullet$ )
$\mathbf{m}'(p) = \mathbf{m}(p) + 1$	if $p \in t^\bullet$ (and $p \notin \bullet t$ )
$\mathbf{m}'(p) = \mathbf{m}(p)$	for all other $p$

  - Notation:  $\mathbf{m} [t] \mathbf{m}'$
- ▶ Marking  $\mathbf{m}$  is *1-bounded* iff  $\mathbf{m}(p) \in \{0, 1\}$  for all  $p$

(... from various places ...)

### Petri nets 3/5

- ▶ Marking  $\mathbf{m} = (1100)$ :
- ▶ Transition  $t_2$  is enabled
  
- ▶ Firing  $t_2$  at  $\mathbf{m}$  leads to  $\mathbf{m}' = (0011)$ :
- ▶ Now  $t_1$  and  $t_3$  are enabled



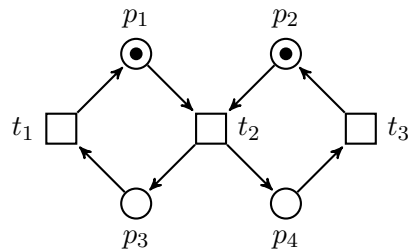
(... from various places ...)

### Petri nets 4/5

- ▶ A pair  $\langle N, \mathbf{m}_0 \rangle$  of a Petri net and an initial marking is called a *marked net*, or *net system*
- ▶ For a marked net  $\mathcal{N} = \langle \langle P, T, F \rangle, \mathbf{m}_0 \rangle$ :
  - A *firing sequence* (or *occurrence sequence*) of  $\mathcal{N}$  is a sequence of transitions in  $T$ ,  $t_1, t_2, \dots, t_n$ , such that  $\mathbf{m}_0 [t_1] \mathbf{m}_1 [t_2] \dots [t_n] \mathbf{m}_n$  for some  $\mathbf{m}_1 \dots \mathbf{m}_n$
  - Notation:  $\mathbf{m}_0 [t_1, \dots, t_n] \mathbf{m}_n$
  - A marking  $\mathbf{m}$  is *reachable* in  $\mathcal{N}$  iff there exists a firing sequence  $t_1 \dots t_n$  of  $\mathcal{N}$  such that  $\mathbf{m}_0 [t_1, \dots, t_n] \mathbf{m}$

(... from various places ...)

## Petri nets 5/5



- ▶ (1001) is reachable via the sequence  $t_2, t_1$  (and also via  $t_2, t_1, t_3, t_2, t_1$ , etc)
- ▶ (1110) is *not* reachable

(... from various places ...)

## 1-Safety 1/2

- ▶ A marked net  $\mathcal{N} = \langle N, \mathbf{m}_0 \rangle$  is **1-safe** iff every reachable marking  $\mathbf{m}$  is 1-bounded ( $\mathbf{m}(p) \in \{0, 1\}, \forall p$ )
  - Places in a 1-safe net may be viewed as propositions (true if marked, false if unmarked)
  - A marking can be given as the set of marked places
- ▶ A Petri net  $N$  is (**structurally**) **1-safe** iff  $\langle N, \mathbf{m}_0 \rangle$  is 1-safe for any 1-bounded initial marking  $\mathbf{m}_0$

(... from various places ...)

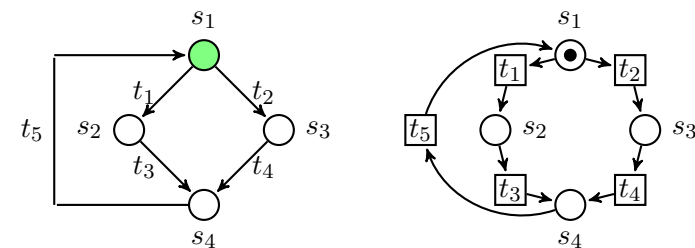
## 1-Safety 2/2

- ▶ Equivalent concept in planning formalisms:
  - STRIPS: an operator is safe if it does not delete any proposition that is already false, or add any proposition that is already true (in any reachable state where the operator is applicable)
  - SAS+: operator  $o$  is safe if whenever  $post(o)[v]$  is defined, so is  $pre(o)[v]$  and  $pre(o)[v] \neq post(o)[v]$

(... from various places ...)

## Petri net representation of transition systems

- ▶ States map to places, transitions to transitions
- ▶ Initial marking marks only the initial state

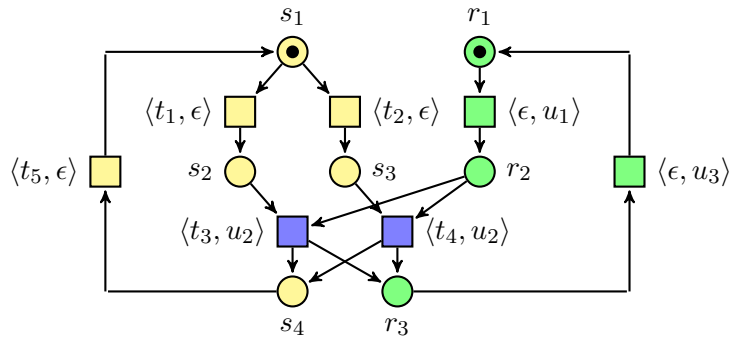


- ▶ The Petri net corresponding to a transition system is inherently 1-safe

(... from various places ...)

## Petri net representation of products 1/2

- ▶ Union of the Petri net representations of product systems
- ▶ Transitions that participate in a synchronisation constraint are “merged”



- ▶ The product net is also 1-safe

(... from various places ...)

ICAPS 2009 25 / 140

## Petri net representation of products 2/2

- ▶ Formally, the marked Petri net representation of the product  $\mathbf{A} = \langle \mathcal{A}_1, \dots, \mathcal{A}_n, \mathbf{T} \rangle$  is  $\langle \langle P, T, F \rangle, \mathbf{m}_0 \rangle$ , where:
  - $P = S_1 \cup S_2 \cup \dots \cup S_n$
  - $T = \mathbf{T}$
  - $F = \{(s, \mathbf{t}) : \exists i. s = \alpha_i(\mathbf{t}_i)\} \cup \{(\mathbf{t}, s) : \exists i. s = \beta_i(\mathbf{t}_i)\}$
  - $\mathbf{m}_0 = \{i s_1, \dots, i s_n\}$

(... from various places ...)

ICAPS 2009 26 / 140

## Some decision problems for marked Petri nets

For a given marked net  $(N, \mathbf{m}_0)$ :

- ▶ **Coverability** of a set of places  $G$ :  
Is there a firing sequence  $s$ , valid at  $\mathbf{m}_0$  that leads to a marking which marks all places in  $G$ ?
- ▶ **Reachability** of a set of places  $G$ :  
Is there a firing sequence  $s$ , valid at  $\mathbf{m}_0$  that marks all *and only* the places in  $G$ ?
- ▶ **Executability** of transition  $t$ :  
Is there a valid firing sequence that contains  $t$ , i.e., can  $t$  ever be executed?

Note: For 1-safe nets, reachability, coverability and executability are all easily reduced to one another.

(... from various places ...)

ICAPS 2009 27 / 140

## Some decision problems for marked Petri nets

- ▶ **Repeated Executability** of transition  $t$ :  
Is there a valid firing sequence that contains  $t$  *infinitely* often?
- ▶ **Livelock**  
Let  $\mathbf{L} \subseteq \mathbf{T}$  be a set of “visible” transitions: Is there an infinite global history in which some transition in  $\mathbf{L}$  occurs, followed by an infinite sequence of invisible transitions?

All the above problems are *PSPACE-Complete* for 1-safe Petri nets, as well as for products of transition systems.

(... from various places ...)

ICAPS 2009 28 / 140

## Petri nets for planning problems

- ▶ Transition systems for each variable extracted from the Domain Transition Graphs (DTGs) of the planning problem
- ▶ Synchronised products formed by taking the global transitions as the (ground) actions in the planning problem

(... from various places ...)

ICAPS 2009 29 / 140

## Planning via Petri nets

- ▶ Plan existence can be decided using Petri nets as follows:
  - Extract the DTGs for each variable  $X$  in the planning problem and make a transition system  $\mathcal{A}_X$
  - Form the synchronised product using as global constraint the actions in the planning problem
  - Create a new global transition  $t_{\text{goal}}$  whose input is the goal of the planning problem and output a **new** place

### Theorem

*There is a valid plan iff  $t_{\text{goal}}$  is executable.*

- ▶ This procedure doesn't compute plans, yet we will come to this issue later...

(... from various places ...)

ICAPS 2009 30 / 140

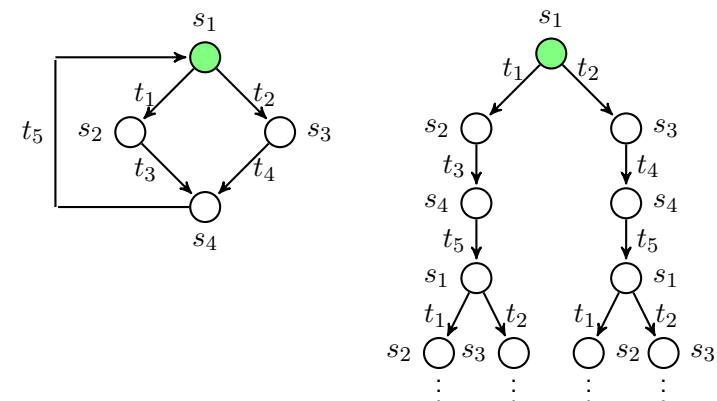
## Part 2: Unfolding

- ▶ Unfolding is an analysis method for 1-safe Petri nets, with interesting and useful properties.
  - Partial-order method: Exploits event concurrency to avoid explosion of interleavings.
  - Can be directed by state-space search heuristics.
- ▶ Using unfolding for planning:
  - Mapping planning problems to 1-safe Petri nets.
  - Properties of generated plans: Concurrency and optimality.

(... from various places ...)

ICAPS 2009 31 / 140

## Unfolding of transition systems 1/2



(... from various places ...)

ICAPS 2009 32 / 140



## Unfolding of transition systems 2/2

- ▶ The unfolding of a transition system is a transition system **with labels**
- ▶ The labels refer to states/transitions of the original transition system
- ▶ States and transitions are called occurrences
- ▶ A state/transition may occur an infinite number of time in the unfolding

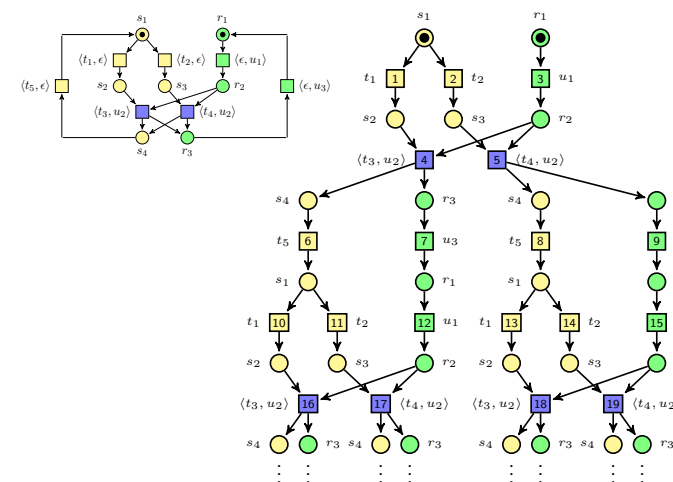
## Unfolding of a product

- ▶ Can unfold the interleaving semantics of a product (need the interleaving semantics of exponential size)
- ▶ Instead, we unfold the Petri net representation of the product
- ▶ For this, we need to define branching processes

## Branching process

- ▶ A branching process is a labeled Petri net that captures the computations of a Petri net
- ▶ When unfolding a Petri net, we start with the places with initial tokens and the net is unfolded iteratively using:
  1. If, in the current net, there is a reachable marking that enables a global transition  $t$ , then a **new transition** labeled by  $t$  and **new places** labeled with the states of  $t^\bullet$  are added to the current net

## Unfolding a product: Example



## Fundamental properties of the unfolding

- ▶ The unfolding is the (unique and perhaps infinite) limiting branching process
- ▶ The unfolding contains all computation histories of the net
- ▶ A marking is reachable in a Petri net iff it “appears” as a marking in the unfolding
- ▶ The unfolding has no cycles and no **backward** conflicts (places with more than one incoming arrow)

(... from various places ...)

ICAPS 2009 37 / 140

## Causality, conflict and concurrency 1/2

- ▶ A node  $x$  (in the unfolding) is a **causal predecessor** of  $y$ , denoted by ' $x < y$ ', if there is a (non-empty) directed path from  $x$  to  $y$
- ▶ Nodes  $x$  and  $y$  are in **conflict**, denoted by ' $x \# y$ ', if there is a place  $z$ , different from  $x$  and  $y$ , from which one can reach  $x$  and  $y$  by exiting  $z$  from different arcs
- ▶ Nodes  $x$  and  $y$  are **concurrent**, denoted by ' $x co y$ ', if  $x$  and  $y$  are neither causally related nor in conflict

(... from various places ...)

ICAPS 2009 38 / 140

## Causality, conflict and concurrency 2/2

### Theorem

*Two nodes  $x$  and  $y$  are either causally related, in conflict, or concurrent.*

### Theorem

*If  $x$  and  $y$  are causally related, then either  $x < y$  or  $y < x$ , but not both.*

### Theorem

*Let  $P$  be a set of places of a branching process  $\mathcal{N}$  of a product  $\mathbf{A}$ . There is a reachable marking  $M$  of  $\mathcal{N}$  such that  $P \subseteq M$  iff the places of  $P$  are pairwise concurrent.*

(... from various places ...)

ICAPS 2009 39 / 140

## Configurations 1/2

- ▶ A **realization** of a set of events is an occurrence sequence (of the branching process) in which every event occurs exactly once, and no other event occurs
- ▶ E.g.,  $\{1, 2\}$  and  $\{4, 6\}$  have no realizations,  $\{1, 3, 4, 7\}$  has the two realizations 1347 and 3147
- ▶ A set of events  $E$  is a **configuration** if it has at least one realization
- ▶ A set of events  $E$  is **causally closed** if  $e \in E$  and  $e' < e$  implies  $e' \in E$

(... from various places ...)

ICAPS 2009 40 / 140

## Configurations 2/2

### Theorem

Let  $E$  be a set of events. Then,

1.  $E$  is a configuration if it is causally closed and no two events in  $E$  are in conflict.
2. All realizations of a finite configuration lead to the same reachable marking.

## Verification using unfoldings

### The question

*Does some computation history execute transition  $t$ ?*

can be answered by exploring the unfolding:

1. compute larger and larger portions of the unfolding until finding an event labeled with  $t$ , or
2. until “somehow” we are able to determine that no further event will be labeled with  $t$

## Constructing the unfolding 1/4

- ▶ Given a branching process  $\mathcal{N}$ , we need to compute the events that extend  $\mathcal{N}$
- ▶ More formally, given  $\mathcal{N}$  and a global transition  $t$ , how can we decide whether  $\mathcal{N}$  can be extended with an event labeled by  $t$ ?
- ▶ Let  $\bullet t = \{s_1, \dots, s_k\}$ . The number  $k$  is the number of components participating in  $t$
- ▶ This number is called the **synchronisation degree** of  $t$

## Constructing the unfolding 2/4

- ▶  $\mathcal{N}$  can be extended with an event labeled by  $t$  iff there is a reachable marking that puts a token on places  $p_1, \dots, p_k$  labeled by  $s_1, \dots, s_k$
- ▶ The following procedure solves this problem:
  1. consider all **candidate sets**  $\{p_1, \dots, p_k\}$  of places of  $\mathcal{N}$  labeled by  $\{s_1, \dots, s_k\}$
  2. for each candidate  $\{p_1, \dots, p_k\}$ , test whether there is a reachable marking  $\mathbf{m}$  that contains  $\{p_1, \dots, p_k\}$ . If so, we say that the candidate is reachable

## Constructing the unfolding 3/4

- ▶ A candidate set is reachable iff its places are pairwise concurrent. This can be checked in  $O(k^2)$  time
- ▶ Therefore, checking whether  $\mathcal{N}$  can be extended with an event labeled  $t$  can be done in time

$$O(n^k/k^k)O(k^2) = O(n^k/k^{k-2})$$

## Constructing the unfolding 4/4

### Theorem

Let  $\mathcal{N}$  be a branching process of a product  $\mathbf{A}$  and  $t$  a global transition. If  $\mathbf{A}$  is of bounded synchronisation degree, then deciding whether  $\mathcal{N}$  can be extended with an event labeled by  $t$  can be done in polynomial time.

### Theorem

In general, deciding whether a branching process can be extended with an event labeled by  $t$  is NP-complete.

## Search procedures (1/2)

We mentioned earlier that we can *somehow* construct larger and larger portions of the unfolding, to answer questions like:

1. Executability (Verification) - does some run contain a particular transition?
2. Repeated executability - does some run contain a particular transition an infinite number of times?
3. Livelock - does some run have an infinite tail of "silent" transitions?

This is done using **search procedures**

## Verification: Search procedures 1/2

Use the unfolding to compute answers to verification questions:

- ▶ Compute more and more of the unfolding, until there is enough information to answer the verification question
- ▶ Use a **search procedure** to compute the unfolding and determine when the question is answered:
  - **search strategy** specifies the event to be added next
  - **search scheme** determines which leaves don't need to be explored further (**termination condition**), and when the search has been successful (**success condition**)

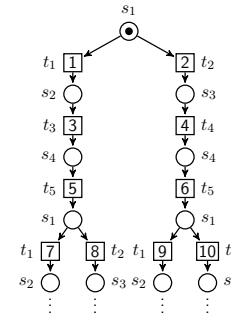
## Verification: Search procedures 2/2

$\mathcal{N}$  := unique branching process without events  
 $T := \emptyset$  /\* terminal events \*/  
 $S := \emptyset$  /\* successful terminals \*/  
 $X := Ext(\mathcal{N}, T)$  /\* possible extensions of  $\mathcal{N}$  \*/  
**while**  $X \neq \emptyset$  **do**  
   Choose an event  $e \in X$  according to search strategy  
   Extend  $\mathcal{N}$  with  $e$   
   **if**  $e$  is terminal according to search scheme **then**  
      $T := T \cup \{e\}$   
     **if**  $e$  is successful according to search scheme **then**  
        $S := S \cup \{e\}$   
     **end if**  
   **end if**  
    $X := Ext(\mathcal{N}, T)$   
**end while**  
**return**  $\langle \mathcal{N}, T, S \rangle$

(... from various places ...)

## Search strategies for transition systems 1/2

- ▶ A strategy selects the next event to add
- ▶ It is a (partial) order on events ... but with care ...
- ▶ We define it as an order on **histories of events**
- ▶  $H(t) = t_1 \dots t_n$  where  $e_1 \dots e_n$  are causal predecessors of  $e$  and  $t_i$  is label of  $e_i$
- ▶ The state reached by  $H(e)$  is  $St(e) = \beta(e)$



$H(7) = t_1 t_3 t_5 t_1$  and  $St(7) = s_2$ .  
 Also,  $H(7) = H(3) t_5 t_1$ .

(... from various places ...)

## Search strategies for transition systems 2/2

- ▶ A search strategy  $\prec$  for transition systems is an order on  $T^*$  that refines the prefix order (i.e.,  $w$  is a proper prefix of  $w'$  then  $w \prec w'$ )
- ▶ Observe that if  $e < e'$  then  $H(e) \prec H(e')$  and thus  $e \prec e'$
- ▶ Therefore, a search strategy **refines the causal order** on events

(... from various places ...)

## Search scheme for transition systems 1/3

Let  $\prec$  be a search strategy. An event  $e$  is **feasible** if no event  $e' < e$  is terminal. A feasible event  $e$  is **terminal** if either

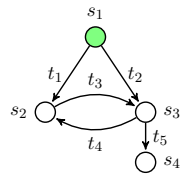
1.  $e$  is labeled with a goal transition (successful terminal), or
2. there is a feasible event  $e' \prec e$  such that  $St(e') = St(e)$

The  $\prec$ -**final prefix** is the prefix of the unfolding containing only feasible events.

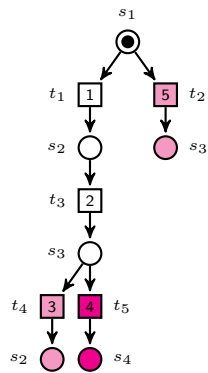
(... from various places ...)

### Search scheme for transition systems 2/3

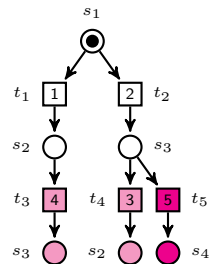
$$G = \{t_5\}$$



$\prec_1 = \text{lexicographic}$



$\prec_2 = \text{smaller } |H(e)|$



(... from various places ...)

### Search scheme for transition systems 3/3

#### Theorem

The search scheme is **sound and complete** for every strategy.

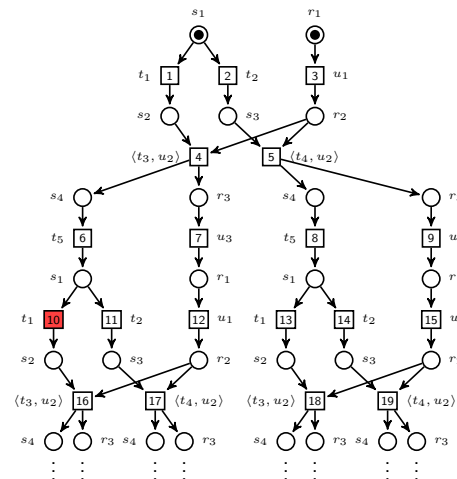
(... from various places ...)

### Search strategies for products 1/3

- ▶ For transition systems, a strategy is an order on  $T^*$  (histories of events)
- ▶ This is possible since every event has a **unique history**
- ▶ Unfortunately, for products, events may have multiple histories

(... from various places ...)

### Search strategies for products 2/3



histories

1	3	4	5	6	12	10
1	3	4	6	10		
3	1	4	6	10		
1	3	4	6	10	7	

(... from various places ...)

## Search strategies for products 3/3

- ▶ So, we are forced to consider subsets of histories ...
- ▶ but we consider those that correspond to **Mazurkiewicz traces**

(... from various places ...)

ICAPS 2009 57 / 140

## Mazurkiewicz traces 1/2

- ▶ Two global transitions are **independent** if no component  $\mathcal{A}_i$  of the product participates in both
- ▶ E.g.,  $\langle t_1, \epsilon \rangle$  and  $\langle \epsilon, u_1 \rangle$  are independent transitions
- ▶ If  $t$  and  $u$  are independent. Then, for  $w, w' \in \mathbf{T}^*$ 
  1. if  $wtuw'$  is a history, then so is  $wutw'$
  2. if  $wt$  and  $wu$  are histories, then so are  $wtu$  and  $wut$

(... from various places ...)

ICAPS 2009 58 / 140

## Mazurkiewicz traces 2/2

- ▶ Two words  $w, w' \in \mathbf{T}^*$  are 1-equivalent, denoted by  $w \equiv_1 w'$  iff  $w = w'$  or there are independent transitions  $t$  and  $u$  such that  $w = w_1 t u w_2$  and  $w' = w_1 u t w_2$
- ▶  $w$  is equivalent to  $w'$  if  $w \equiv w'$  where  $\equiv$  is the transitive closure of  $\equiv_1$
- ▶ A (Mazurkiewicz) trace is an equivalence class of  $\equiv$ . The trace of  $w$  is  $[w]$ . A trace is a history trace if all its elements are histories

(... from various places ...)

ICAPS 2009 59 / 140

## Search strategies as orders on traces

- ▶ We follow the same steps as for transition systems:
  1. First, define the set of histories for an event
  2. Show that this set is a trace
  3. Define a strategy as an order on traces
- ▶ The **past** of event  $e$ , denoted by  $past(e)$ , is the set of events  $e'$  such that  $e' \leq e$ ;  $past(e)$  is a configuration
- ▶ A word  $t_1 \dots t_n$  is a history of configuration  $C$  if there is a realization  $e_1 \dots e_n$  of  $C$  such that  $e_i$  is labeled by  $t_i$ . The set of histories of  $C$  is denoted by  $\mathbf{H}(C)$ . The set of histories of  $past(e)$  is denoted by  $\mathbf{H}(e)$ .
- ▶  $\mathbf{H}(C)$  is a trace
- ▶ A strategy is an order on traces that refines the prefix order

(... from various places ...)

ICAPS 2009 60 / 140

## Search scheme for products 1/2

- ▶ Let  $C$  be a configuration. The state reached by  $C$ , denoted by  $\text{St}(C)$ , is the state reached by the execution of any of the histories of  $\mathbf{H}(C)$

Let  $\prec$  be a search strategy. An event  $e$  is **feasible** if no event  $e' \prec e$  is terminal. A feasible event  $e$  is **terminal** if either

1.  $e$  is labeled with a transition of  $\mathbf{G}$  (successful terminal), or
2. there is a feasible event  $e' \prec e$  such that  $\text{St}(e') = \text{St}(e)$

### Theorem

The search scheme is **sound** for every strategy. Unfortunately, it is not complete for every strategy.

(... from various places ...)

ICAPS 2009 61 / 140

## Search schemes for products 2/2

- ▶ A strategy  $\prec$  is **adequate** if
  1. It is well founded
  2. It is preserved by extensions: for all traces  $[\mathbf{w}], [\mathbf{w}'], [\mathbf{w}'']$ , if  $[\mathbf{w}] \prec [\mathbf{w}']$  then  $[\mathbf{w}\mathbf{w}''] \prec [\mathbf{w}'\mathbf{w}'']$

### Theorem

The search scheme is **complete** for all adequate strategies.

### Theorem

The final  $\prec$ -prefix has at most  $K$  non-terminal nodes if  $\prec$  is a total order where  $K$  is the number of global states.

(... from various places ...)

ICAPS 2009 62 / 140

## The size and Parikh strategies

- ▶ The size strategy, denoted  $\prec_s$ , is:  $[\mathbf{w}] \prec_s [\mathbf{w}']$  if  $|\mathbf{w}| < |\mathbf{w}'|$
- ▶ The **Parikh mapping** of  $\mathbf{w}$  is the function  $\mathcal{P}([\mathbf{w}])$  that maps each transition  $\mathbf{t}$  to the number of times it occurs in  $\mathbf{w}$
- ▶ Given a total order  $<_a$  on transitions. The Parikh strategy, denoted  $\prec_P$ , is:  $[\mathbf{w}] \prec_P [\mathbf{w}']$  if  $[\mathbf{w}] \prec_s [\mathbf{w}']$ , or  $[\mathbf{w}] =_s [\mathbf{w}']$  and there is  $\mathbf{t}$  such that
  1.  $\mathcal{P}([\mathbf{w}])(\mathbf{t}) < \mathcal{P}([\mathbf{w}'])(\mathbf{t})$  and
  2.  $\mathcal{P}([\mathbf{w}])(\mathbf{t}') = \mathcal{P}([\mathbf{w}'])(\mathbf{t}')$  for every  $\mathbf{t}' <_a \mathbf{t}$
- ▶ The size and Parikh strategies are adequate but not total
- ▶ There are other (more complex) total and adequate strategies

(... from various places ...)

ICAPS 2009 63 / 140

## Directed unfolding

- ▶ In the verification problem, we search the unfolding for an event labeled by a transition in  $\mathbf{G}$  until we find it or conclude no such event exists
- ▶ In directed unfolding, we guide the search with a heuristic function that estimates how far the desired event is from a given part of the branching process
- ▶ It is the same idea used in heuristic search in which instead of making a blind search, a heuristic function is used to focus the search
- ▶ As expected, when the target event is reachable, directed unfolding is order of magnitude more efficient than “blind” unfolding

(... from various places ...)

ICAPS 2009 64 / 140



## Heuristic-guided strategies 1/2

Let  $C$  be a configuration

- ▶ Define  $g(C)$  as the size  $|C|$
- ▶ Let  $h$  map configurations  $C$  into reals  $[0, \infty]$  such that
  1. if  $\mathbf{St}(C) = \mathbf{St}(C')$  then  $h(C) = h(C')$
  2. if  $\mathbf{H}(C) \cap \mathbf{G} \neq \emptyset$ , then  $h(C) = 0$
- ▶ Define  $f(C) = g(C) + h(C)$

(... from various places ...)

ICAPS 2009 65 / 140

## Heuristic-guided strategies 2/2

- ▶ Define the order  $\prec_h$  on histories as follows:

$$[\mathbf{w}] \prec_h [\mathbf{w}'] \text{ iff } \begin{cases} f([\mathbf{w}]) < f([\mathbf{w}']) & \text{if } f([\mathbf{w}]) < \infty \\ |\mathbf{w}| < |\mathbf{w}'| & \text{if } f([\mathbf{w}]) = f([\mathbf{w}']) = \infty \end{cases}$$

### Theorem

The  $\prec_h$ -final prefix is finite, and the search scheme is sound and complete.

- ▶ The strategy  $\prec_h$  is a  $h$ -focused strategy

(... from various places ...)

ICAPS 2009 66 / 140

## Heuristics

- ▶ By definition,  $h$  maps global states into non-negative numbers
- ▶ Therefore, we can use any heuristic function defined on global states such as
  1.  $h_{max}$
  2.  $h_{add}$
  3.  $h_{FF}$
  4. etc

(... from various places ...)

ICAPS 2009 67 / 140

## Experimental results 1/2

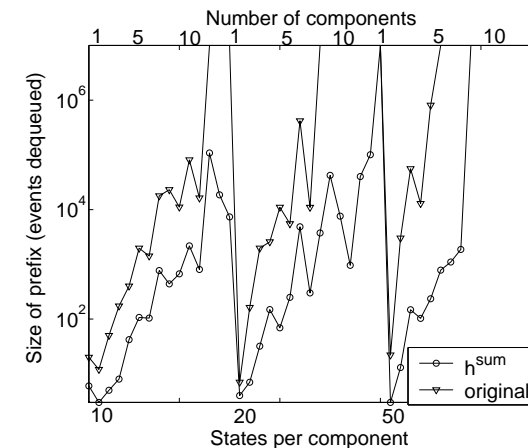


Figure: Results for Random PT-nets: Node Expansions

(... from various places ...)

ICAPS 2009 68 / 140

## Experimental results 2/2

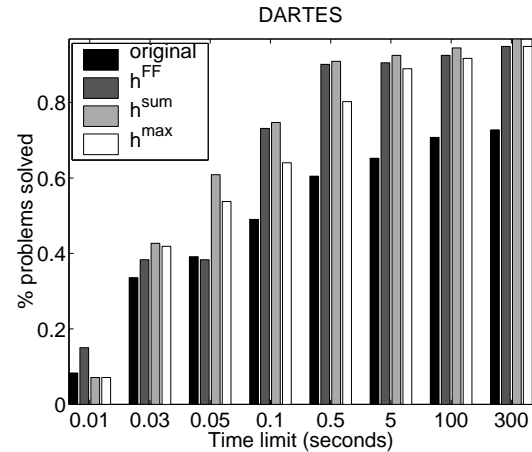


Figure: Results for Dartes: Node Expansions

(... from various places ...)

ICAPS 2009 69 / 140

## Planning Problem

Denote a planning problem by  $\mathcal{P} = \langle V, O, S_0, G \rangle$ , where

- ▶  $V$  is a set of multi-valued state variables
- ▶  $O$  is a set of (grounded) operators characterised by their pre and post conditions.
- ▶  $S_0$  is the fully specified initial state
- ▶  $G$  is the fully or partially specified goal state

(... from various places ...)

ICAPS 2009 70 / 140

## Partially-Ordered Plan

- ▶ A partially-ordered plan  $\pi = \langle A, < \rangle$  consists of a multiset of operators  $A$  in  $O$  and a strict partial order relation  $<$  over  $A$ .
- ▶  $\pi$  is a solution plan for planning problem  $\mathcal{P}$  if any linearization of  $\pi$  will transition the system from  $S_0$  to a state where all goal propositions hold.

(... from various places ...)

ICAPS 2009 71 / 140

## Using Unfolding for Planning (1/4)

1. Cast planning problem to Petri net executability problem
2. Unfold to solve the related executability question
3. Extract plan

(... from various places ...)

ICAPS 2009 72 / 140

## Using Unfolding for Planning (2/4)

### 1. Cast planning problem to Petri net executability problem

- 1.1 Map  $O$  to a set of 1-safe operators  $O'$ .
- 1.2 For a STRIPS problem where  $V$  is a set of propositions, introduce complementary set  $\hat{V}$  and replace every instance of  $\neg v$  with  $\hat{v}$ .
- 1.3 For each variable  $X \in V$  extract the DTG and make a transition system  $\mathcal{A}_X$
- 1.4 Form the synchronised product  $\mathbf{A} = \langle \mathcal{A}_1, \dots, \mathcal{A}_n, \mathbf{T} \rangle$ 
  - Synchronisation constraints  $\mathbf{T}$  are defined by the planning operators  $O'$ .
- 1.5 Map  $\mathbf{A}$  to a Petri net and extend with "goal" transition.
- 1.6 Capture dynamics of prevail conditions.

(... from various places ...)

ICAPS 2009 73 / 140

## 1-Safe Operators (1/2)

- ▶ Recall: SAS+ operator  $o$  is safe iff whenever  $post(o)[v]$  is defined, so is  $pre(o)[v]$  and  $pre(o)[v] \neq post(o)[v]$
- ▶ Translating a non-safe operator:

door: open, closed

shut-door =  $\langle \{at-door\}, \{closed\} \rangle$

shut-door1 =  $\langle \{at-door, closed\}, \{ \} \rangle$

shut-door2 =  $\langle \{at-door, open\}, \{closed\} \rangle$

- ▶ Number of copies created is exponential in the number of missing preconditions

(... from various places ...)

ICAPS 2009 74 / 140

## 1-Safe Operators (2/2)

- ▶ Operator may be safe, without satisfying the definition, due to mutexes between values of different variables.
- ▶ Use standard reachability analysis techniques to identify such cases (computing mutexes and state invariants, as in e.g. [Bonet & Geffner '99, Helmert '06])
- ▶ Many of the standard benchmark domains are already 1-safe, or nearly 1-safe.

(... from various places ...)

ICAPS 2009 75 / 140

## Product of State Variable DTGs

- ▶ For each variable  $X \in V$  extract the DTG and make a transition system  $\mathcal{A}_X$
- ▶ Form the synchronised product  $\mathbf{A} = \langle \mathcal{A}_1, \dots, \mathcal{A}_n, \mathbf{T} \rangle$
- ▶ Synchronisation constraints  $\mathbf{T}$  are defined by the 1-safe planning operators.

(... from various places ...)

ICAPS 2009 76 / 140

## Petri net representation

- ▶ Build the marked Petri net representation of  $\mathbf{A}$ , as described previously.
- ▶ Create a new transition  $t_{goal}$  whose input is the goal of the planning problem and output is a **new** place

(... from various places ...)

## Translate Prevail conditions (1/3)

**Problem:** Two actions with a common prevail condition will be prohibited from executing concurrently.

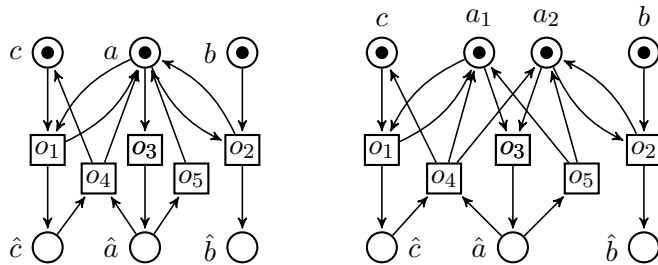
Let  $a_1, a_2$  be two actions with common prevail condition  $p$

- ▶ Any two events  $e_1$  and  $e_2$  in the unfolding, labeled by  $a_1$  and  $a_2$  respectively, will be in conflict due to  $p$ , i.e.  $e_1 \# e_2$ .
- ▶ Any plan containing  $a_1$  and  $a_2$  will necessarily require that  $a_1 < a_2$  or  $a_2 < a_1$ .

(... from various places ...)

## Translate Prevail Conditions (2/3)

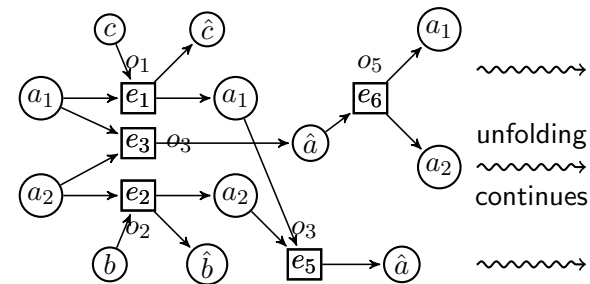
To overcome this we can apply the place replication technique proposed by [Vogler, Semenov, Yakovlex, 1998]



Picture by Sebastian Sardina

(... from various places ...)

## Translate Prevail Conditions (3/3)



Picture by Sebastian Sardina

- ▶ Denote the Petri net representation of planning problem  $\mathcal{P}$  as  $\mathcal{N}_{\mathcal{P}}$ .

(... from various places ...)

## Using Unfolding for Planning (3/4)

1. Cast planning problem  $\mathcal{P}$  to Petri net executability problem
2. **Unfold  $\mathcal{N}_{\mathcal{P}}$  to solve the related executability problem**
  - Is the transition  $t_g$  executable?
  - Direct the unfolding using a sound and complete scheme
  - May choose to use planning heuristic, etc.
  - Denote to the resulting final prefix as  $Unf_{\prec}(\mathcal{N}_{\mathcal{P}})$
3. Extract plan

(... from various places ...)

ICAPS 2009 81 / 140

## Using Unfolding for Planning (4/4)

1. Cast planning problem to Petri net executability problem
2. Unfold to solve the related executability problem
3. **Extract plan from  $Unf_{\prec}(\mathcal{N}_{\mathcal{P}})$** 
  - (Assuming success)
  - Linear time
  - Solution plan  $\pi = \langle H(e_g), \prec \rangle$  where  $e_g$  is an event in  $Unf_{\prec}(\mathcal{N}_{\mathcal{P}})$  labeled by  $t_g$  and  $\prec$  is the finite closure of the causal relation over  $H(e_g)$ .
  - E.g.  $\pi = \langle \{ o1, o2, o3 \}, \{ o1 \prec o3 \} \rangle$
  - True concurrency semantics
  - E.g.  $o2$  can temporally overlap with  $o1$  and  $o3$

(... from various places ...)

ICAPS 2009 82 / 140

## Plan generated via unfolding

### Theorem

*A plan  $\pi$ , extracted from  $Unf_{\prec}(\mathcal{N}_{\mathcal{P}})$  as described, is a solution plan for planning problem  $\mathcal{P}$*

- ▶ Let us refer to this simply as a plan obtained via unfolding.

(... from various places ...)

ICAPS 2009 83 / 140

## Concurrency Semantics

- ▶ What is the concurrency semantics of plans synthesised using this approach?
  - What are the restrictions on two actions executing concurrently?
- ▶ How does it compare to the standard notion of concurrency induced by Smith and Weld's [1999] definition of independent actions?

(... from various places ...)

ICAPS 2009 84 / 140

## Independent Actions

Two actions are **independent** iff

1. Their effects don't contradict
2. Their preconditions don't contradict
3. The preconditions for one aren't clobbered by the effect of the other.

A plan **respects independence** iff for any two non-independent actions  $a$  and  $b$  the plan ensures that either  $a < b$  or  $b < a$ .

- ▶ Obviously any totally ordered plan respects independence

### Theorem

*A plan generated via unfolding respects independence.*

## Stronger than Independence...

Moreover, planning via unfolding enforces **stronger** restrictions on when two actions can be executed concurrently:

- ▶ Operators in  $O$  with common postcondition  $v = v_1$  can't temporally overlap if their common effect changes the current state.
- ▶ Occurs through 1-safety transformation of operators
  - Value of  $v$  not specified in the preconditions
  - Create set of operators to specify the value of  $v$  in the preconditions,
  - May be that original operators are independent but translated ones are not.

## Strongly Independent Actions

Two actions are **strongly independent** in state  $S$  iff

1. They are independent
2. Any postcondition  $p$  common to both actions already holds true in  $S$ .

- ▶ A variable is locked in shared mode if the action does not change its value (read only access)
- ▶ A variable is locked in exclusive mode if its value is to be changed by the action (read and write access)

Strong independence reduces to independence if operators are originally 1-safe.

## Strongly Independent Plan

- ▶ Let  $\text{state}(\pi, S_0, a)$  denote the set of states in which action  $a$  may potentially be executed when a linearisation of plan  $\pi$  is executed in state  $S_0$ .

A **plan**  $\pi$  respects **strong independence** for state  $S_0$  iff

- ▶ For any two different action (instances)  $a$  and  $b$  which are not strongly independent for some state  $S \in \text{state}(\pi, S_0, a)$ , the plan ensures that either  $a < b$  or  $b < a$

## Unfolding Synthesizes Strongly Independent Plan

### Theorem

*A plan generated via unfolding respects strong independence for the initial state of the planning problem.*

- ▶ Are solution plans **over-constrained** wrt these restrictions?
  - Any totally ordered plan will respect strong independence.

(... from various places ...)

ICAPS 2009    89 / 140

## Plan Flexibility

Partially-ordered plans are in principle more **flexible** in that they may avoid over-committing to action orderings

- ▶ Scheduler can have alternative execution realizations to choose from
  - Sequences in the case of interleaved concurrency
  - Scheduler may be used to post-process or adapt a plan for actions with deadlines and earliest release times
- ▶ Execution time may be reduced when actions can be executed in parallel

(... from various places ...)

ICAPS 2009    90 / 140

## Plan De/reordering

Can we remove (deorder) or change (reorder) the constraints from a plan synthesized via the unfolding approach?

catch-train < cook-dinner < eat-dinner < read-paper

⇓ **deorder** - remove constraints

catch-train < cook-dinner < {eat-dinner, read-paper}

⇕ **reorder** - change constraints

{catch-train, read paper} < cook-dinner < eat-dinner

(... from various places ...)

ICAPS 2009    91 / 140

## Plan validity w.r.t. Strong Independence

A partially ordered plan  $\pi$  is  $\mathcal{P}$ -valid for planning problem  $\mathcal{P}$  iff

- ▶ All linearizations of  $\pi$  solve  $\mathcal{P}$ , and
- ▶  $\pi$  respects strong independence for the initial state of  $\mathcal{P}$ .

### Theorem

*Plans synthesized via the unfolding approach are  $\mathcal{P}$ -valid.*

(... from various places ...)

ICAPS 2009    92 / 140

## Minimal De/re-ordering

Consider plan  $\pi$  which is  $\mathcal{P}$ -valid:

- ▶  $\pi$  is a **minimal de/re-ordering wrt flexibility** if you can't de/re-order it to reduce the number of constraints and retain  $\mathcal{P}$ -validity.
- ▶ A plan is a **minimal de/re-ordering wrt execution time** if you can't de/re-order it to reduce the execution time and retain  $\mathcal{P}$ -validity.

[Backstrom 98] gave similar definitions in the context of plans which respect independence.

## Optimality Guarantees (1/2)

### Theorem

*Any plan synthesized via the unfolding approach is a minimal deordering wrt flexibility.*

- ▶ i.e. no constraint can be removed without rendering the plan invalid.
- ▶ Observe that a minimal deordering wrt flexibility is also a minimal deordering wrt execution time.

### Theorem

*All solution plans which are minimally deordered wrt flexibility exist in the unfolding space.*

- ▶ These results extend to all plans in the unfolding space (i.e. not necessarily solutions)

## Directing the unfolding wrt time

Briefly...

- ▶ Use a search strategy based on an order on events  $\prec_{time}$  that prefers histories with a faster execution time.
- ▶ Use a search scheme based on a semi-admissible order on events  $\prec$ , such that  $\prec_{time} \Rightarrow \prec$
- ▶ i.e. Direct the search using  $\prec_{time}$ , but test termination condition using  $\prec$ .
- ▶ This search procedure will find the fastest plan *in the unfolding space*, but what does this mean?

## Optimality Guarantees (2/2)

### Theorem

*If the unfolding is directed to prefer faster plans, then the plan synthesized is a minimal reordering wrt execution time.*

- ▶ Reordering a plan to be optimal wrt execution time is (still) NP-hard in the context of strong independence requirements.



## Flexibility and Minimal Makespan

So how does planning via unfolding compare to the standard notion of concurrency induced by Smith and Weld's [1999] definition of independent actions?

- ▶ If the original operators are 1-safe then the unfolding space consists of plans which are least-constrained wrt the standard definition of independence.
- ▶ This means a plan with minimum makespan, as defined by Smith and Weld [1999], exists in the unfolding space and can be obtained using an appropriate search procedure.
- ▶ If the operators are not 1-safe, then the unfolding space may contain "slower" (over-constrained) plans due to the stronger restriction on when two actions can temporally overlap.
- ▶ We can guarantee, however, that these plans will be least-constrained wrt strong independence.

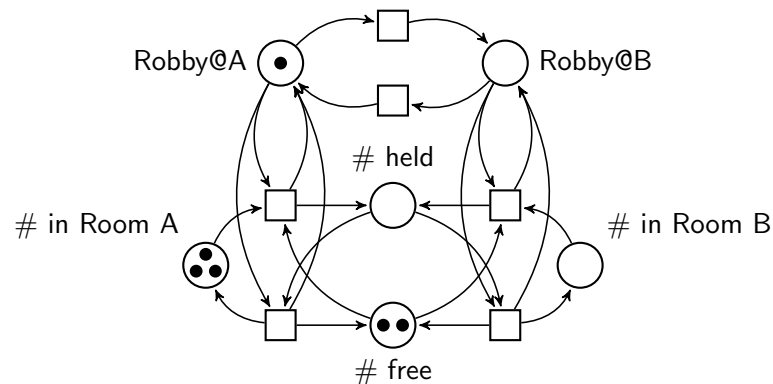
## Part 3: General Petri Nets

- ▶ In general (not 1-safe) Petri nets, places are *unbounded* counters.
  - Petri nets have advantages in expressivity and modelling convenience.
  - Questions of reachability, coverability, etc. are computationally harder to answer, but still decidable.
- ▶ Analysis methods for general Petri nets are often based on ideas & techniques not common in planning:
  - Algebraic methods based on the state equation.
  - Rich literature on the study of classes of nets with special structure.

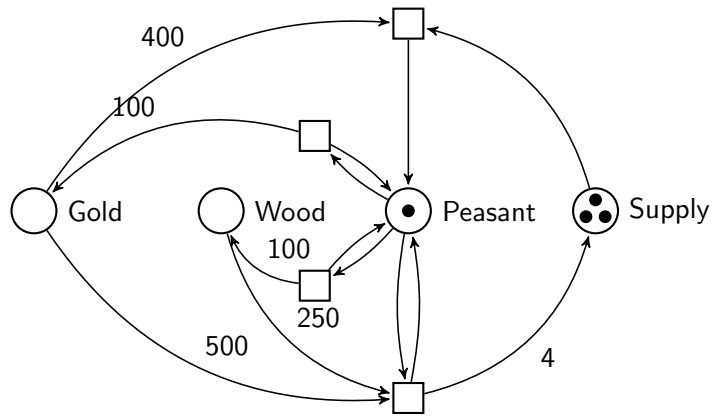
## Recap: Petri Nets

- ▶ A Petri net is a directed bipartite (multi-)graph, with nodes  $P \cup T$  divided into *places* and *transitions*.
  - $F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$  denotes edge *multiplicity*.
  - As usual, for any  $n \in P \cup T$ ,  $\bullet n = \{n' \mid F(n', n) > 0\}$  and  $n^\bullet = \{n' \mid F(n, n') > 0\}$ .
- ▶ A *marking* of the net is a mapping  $P \rightarrow \mathbb{N}$ , i.e., places are *unbounded counters*.
- ▶ A transition  $t$  is *enabled*, or *firable*, at marking  $\mathbf{m}$  iff  $\mathbf{m}[i] \geq F(p_i, t)$ ,  $\forall i$ , and when fired leads to a marking  $\mathbf{m}'$  such that  $\mathbf{m}'[i] = \mathbf{m}[i] - F(p_i, t) + F(t, p_i)$ ,  $\forall i$ .
  - Notation:  $\mathbf{m} [t] \mathbf{m}'$  ( $t$  enabled at  $\mathbf{m}$ :  $\mathbf{m} [t]$ ).
- ▶  $\mathbf{m}_0 [t_1] \mathbf{m}_1 [t_2] \mathbf{m}_2 \dots [t_n] \mathbf{m}_n$  is a (valid) *firing sequence*.
  - Notation:  $\mathbf{m} [t_1, t_2, \dots, t_n] \mathbf{m}_n$ .

## Modelling Planning Problems Using Counters



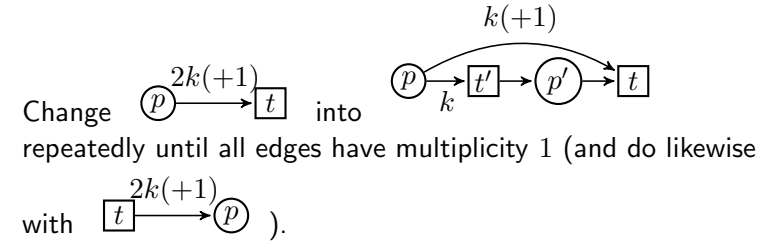
Gripper without Symmetries



Part of the Wargus Domain (Chan et al. 2007)

## Ordinary Petri Nets

- ▶ A Petri net where all edges have multiplicity 1 is *ordinary*.
- ▶ Any net can be transformed into an equivalent ordinary net:



- ▶ The transformation increases net size by  $O(\log(F(p, t)))$ , and hence is linear space.

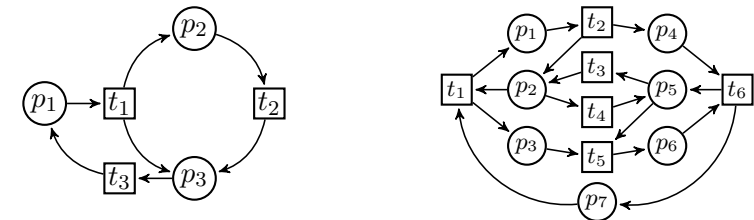
## Vector Notation for Nets and Markings

- ▶ Marking:  $|P|$ -dimensional vector  $\mathbf{m} \in \mathbb{N}^{|P|}$ .
- ▶ **Definition:**  $\mathbf{m} \geq \mathbf{m}'$  iff  $\mathbf{m}[i] \geq \mathbf{m}'[i], \forall i$ .  
 $\mathbf{m} > \mathbf{m}'$  iff  $\mathbf{m} \geq \mathbf{m}'$  and  $\exists j$  such that  $\mathbf{m}[j] > \mathbf{m}'[j]$ .
- ▶ Two vectors associated with transition  $t$ :

$$\mathbf{c}^-(t) = \begin{pmatrix} F(p_1, t) \\ \vdots \\ F(p_{|P|}, t) \end{pmatrix} \quad \mathbf{c}^+(t) = \begin{pmatrix} F(t, p_1) \\ \vdots \\ F(t, p_{|P|}) \end{pmatrix}$$

- $t$  is enabled at  $\mathbf{m}$  iff  $\mathbf{m} \geq \mathbf{c}^-(t)$ ;
- $\mathbf{c}(t) = \mathbf{c}^+(t) - \mathbf{c}^-(t)$  is the effect of  $t$ : firing  $t$  leads to  $\mathbf{m}' = \mathbf{m} + \mathbf{c}(t)$ .
- ▶  $C = (\mathbf{c}(t_1) \mathbf{c}(t_2) \dots \mathbf{c}(t_{|T|}))$  is the *incidence matrix*.
- ▶  $\mathbf{r}(p)$ : row of  $C$  corresponding to place  $p$ .

## Examples

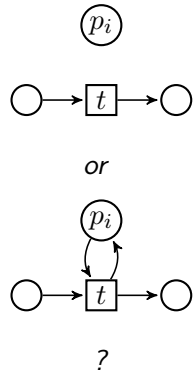


$$C = \begin{pmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 1 & -1 \end{pmatrix}$$

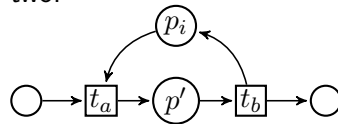
$$C = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

## Representation Ambiguity and Pure Nets

$$c(t)[i] = 0:$$



- ▶ A *pure* Petri net has no “self loops”, i.e.,  $\bullet t \cap t \bullet = \emptyset$  for every transition  $t$ .
- ▶ For a pure net, the incidence matrix  $C$  unambiguously defines the net.
- ▶ Any net can be transformed into a pure net by splitting transitions with loops in two:



- ▶ The transformation is linear space.

(... from various places ...)

ICAPS 2009 105 / 140

## Decision Problems for Marked Nets

- ▶ Given a marked net  $(N, \mathbf{m}_0)$ :
  - **Reachability:** Is there a firing sequence that ends with given marking  $\mathbf{m}$ ?
  - **Coverability:** Is there a firing sequence that ends with a marking  $\mathbf{m}'$  such that  $\mathbf{m}' \geq \mathbf{m}$ ?
  - **Boundedness:** Does there exist a (finite)  $\mathbf{K}$  such that for every reachable marking  $\mathbf{m}$ ,  $\mathbf{m} \leq \mathbf{K}$ ?
    - Note: The state space of  $(N, \mathbf{m}_0)$  is finite iff  $(N, \mathbf{m}_0)$  is bounded.
- ▶ Coverability and boundedness are EXPSPACE-complete.
- ▶ Reachability is EXPSPACE-hard, but existing algorithms are non-primitive recursive (i.e., have unbounded complexity).

(... from various places ...)

ICAPS 2009 106 / 140

- ▶ **Executability:** Is there a firing sequence valid at  $\mathbf{m}_0$  that includes transition  $t$ ?
  - Executability reduces to coverability:  $t$  is executable iff  $c^-(t)$  is coverable.
  - and vice versa: reduction using a “goal transition”.
- ▶ **Repeated Executability:** Is there a firing sequence in which a given transition (or set of transitions) occurs an infinite number of times?
- ▶ **Reachable Deadlock:** Is there a reachable marking  $\mathbf{m}$  at which no transition is enabled?
- ▶ **Liveness:** Absence of reachable deadlocks.
- ▶ ...and many more (e.g., existence of home states, fairness).

(... from various places ...)

ICAPS 2009 107 / 140

## Equivalence Problems

- ▶ **Equivalence:** Given two different marked nets,  $(N_1, \mathbf{m}_1)$  and  $(N_2, \mathbf{m}_2)$ , with equal (or isomorphic) sets of places, do they have the equal sets of reachable markings?
- ▶ **Trace Equivalence:** Given two different marked nets,  $(N_1, \mathbf{m}_1)$  and  $(N_2, \mathbf{m}_2)$ , with equal (or isomorphic) sets of transitions, do they have equal sets of valid firing sequences?
- ▶ **Language Equivalence:** Trace equivalence under mapping of transitions to a common alphabet.
- ▶ **Bisimulation:** Equivalence under a bijection between markings.
- ▶ In general, equivalence problems are undecidable.

(... from various places ...)

ICAPS 2009 108 / 140

## Structural Properties & Decision Problems

- ▶ Properties of  $N$  independent of initial marking  $\mathbf{m}_0$ .
- ▶ **Invariance:**
  - A vector  $\mathbf{y} \in \mathbb{N}^{|P|}$  is a *P-invariant* of  $N$  iff for any markings  $\mathbf{m} [\cdot \cdot \cdot] \mathbf{m}'$ ,  $\mathbf{y}^T \mathbf{m} = \mathbf{y}^T \mathbf{m}'$ .
  - A P-invariant is a linear combination of place markings that is invariant under any transition firing.
  - In Germany, *S-invariant*; also called *P-semiflow*.
  - A vector  $\mathbf{x} \in \mathbb{N}^{|T|}$  is a *T-invariant* of  $N$  iff for any firing sequence  $s$  such that  $\mathbf{n}(s) = \mathbf{x}$  and any marking  $\mathbf{m}$  where  $s$  is enabled,  $\mathbf{m} [s] \mathbf{m}$ .
  - A T-invariant is a multiset of transitions whose combined effect is zero.

(... from various places ...)

ICAPS 2009 109 / 140

- ▶ **Structural Liveness:** Is there a marking  $\mathbf{m}$  such that  $(N, \mathbf{m})$  is live?
- ▶ **Structural Boundedness:** Is  $(N, \mathbf{m})$  bounded for every finite initial marking  $\mathbf{m}$ ?
- ▶ **Repetitiveness:** Is there a marking  $\mathbf{m}$  and a firing sequence  $s$  valid at  $\mathbf{m}$  such that a given transition / set of transitions appears infinitely often in  $s$ ?
- ▶ Deciding structural properties can be easier than the corresponding problem for a marked net.

(... from various places ...)

ICAPS 2009 110 / 140

## Complexity: Implications Of and For Expressivity

- ▶ Bounded Petri nets are expressively equivalent to propositional STRIPS/PDDL.
  - Reachability is PSPACE-complete for both.
  - Note: The “direct” STRIPS→PN translation can blow up exponentially.
- ▶ General Petri nets are strictly more expressive than propositional STRIPS/PDDL.
- ▶ General Petri nets are at least as expressive as “lifted” (finite 1st order) STRIPS/PDDL.
  - Probably also strictly more expressive, but no proof yet.

(... from various places ...)

ICAPS 2009 111 / 140

- ▶ A *k-counter machine* ( $k\text{CM}$ ) is a deterministic finite automaton with  $k$  (positive) integer counters.
  - Can increment/decrement (by 1), or reset, a counter.
  - Conditional jumps on  $c_i > 0$  or  $c_i = 0$ .
- ▶ Note the differences:
  - A  $k\text{CM}$  is *deterministic*: starting configuration determines a unique execution; a Petri net has choice.
  - A  $k\text{CM}$  can *branch* on  $c_i > 0/c_i = 0$ ; a Petri net can only precondition transitions on  $\mathbf{m}(p_i) > 0$ .
- ▶ A  $k\text{CM}$  is *M-bounded* iff no counter ever exceeds  $M$ .

(... from various places ...)

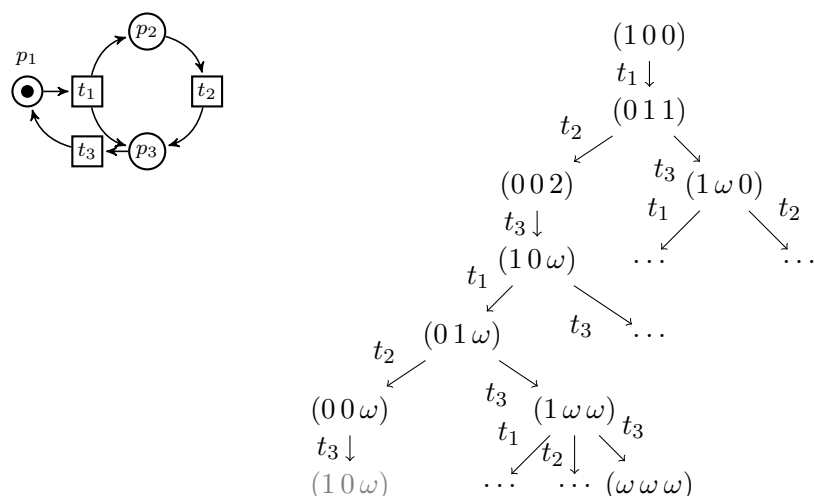
ICAPS 2009 112 / 140

- ▶ An  $n$ -size Turing machine can be simulated by an  $O(n)$ -size 2CM (if properly initialised).
  - Halting (i.e., reachability) for unbounded 2CMs is undecidable.
  - Petri nets are strictly less expressive than unbounded 2CMs.
- ▶ An  $n$ -size and  $2^n$  space bounded TM can be simulated by an  $O(n)$ -size  $2^{2^n}$ -bounded 2CM.
- ▶ A  $2^{2^n}$ -bounded  $n$ -size 2CM can be (non-deterministically!) simulated by an  $O(n^2)$ -size Petri net.
  - Reachability for Petri nets is  $DSPACE(2^{\sqrt{n}})$ -hard.

## The Coverability Tree Construction

- ▶ The *coverability tree* of a marked net  $(N, \mathbf{m}_0)$  is an explicit representation of reachable markings – but not *exactly* the set of reachable markings.
- ▶ Constructed by forwards exploration:
  - Each enabled transition generates a successor marking.
  - If reach  $\mathbf{m}$  such that  $\mathbf{m} > \mathbf{m}'$  for some ancestor  $\mathbf{m}'$  of  $\mathbf{m}$ , replace  $\mathbf{m}[i]$  by  $\omega$  for all  $i$  s.t.  $\mathbf{m}[i] > \mathbf{m}'[i]$ .
    - $\mathbf{m}'[s = t_1, \dots, t_l] \mathbf{m}$ , and since  $\mathbf{m} \geq \mathbf{m}'$ ,  $\mathbf{m}[s] \mathbf{m}''$  such that  $\mathbf{m}'' \geq \mathbf{m}$ ; sequence  $s$  can be repeated any number of times.
    - $\omega$  means “arbitrarily large”.
  - Also check for regular loops ( $\mathbf{m} = \mathbf{m}'$  for some ancestor  $\mathbf{m}'$  of  $\mathbf{m}$ ).
- ▶ Every branch has finite depth.

## Example



## Uses For The Coverability Tree

- ▶ Decides coverability:
  - $\mathbf{m}$  is coverable iff  $\mathbf{m} \leq \mathbf{m}'$  for some  $\mathbf{m}'$  in the tree (where  $n < \omega$  for any  $n \in \mathbb{N}$ ).
  - If  $\mathbf{m}$  is coverable, there exists a covering sequence of length at most  $O(2^n)$ .
- ▶ Decides boundedness:
  - $(N, \mathbf{m}_0)$  is unbounded iff there exists a self-covering sequence:  $\mathbf{m}_0 [s] \mathbf{m} [s'] \mathbf{m}'$  such that  $\mathbf{m}' > \mathbf{m}$ .
  - I.e.,  $(N, \mathbf{m}_0)$  is unbounded iff  $\omega$  appears in some marking in the coverability tree.
  - If  $(N, \mathbf{m}_0)$  is unbounded, there exists a self-covering sequence of length at most  $O(2^n)$ .
- ▶ In general, does *not* decide reachability.
  - Except if  $(N, \mathbf{m}_0)$  is bounded.



- ▶ The *support* of a P-/T-invariant  $\mathbf{y}$  is the set  $\{i \mid \mathbf{y}[i] > 0\}$ . An invariant has *minimal support* iff no invariants support is a strict subset.
  - The number of minimal support P-/T-invariants of a net is finite, but may be exponential.
  - All P-/T-invariants are (positive) linear combinations of minimal support P-/T-invariants.
- ▶ A P-/T-invariant  $\mathbf{y}$  is *minimal* iff no  $\mathbf{y}' < \mathbf{y}$  is invariant.
  - A minimal invariant need not have minimal support.
  - For each minimal support, there is a unique minimal invariant.
- ▶ Algorithms exist to generate all minimal support P-/T-invariants of a net.

(... from various places ...)

ICAPS 2009 121 / 140

## The State Equation & Structural Properties

- ▶  $N$  is structurally bounded iff  $\mathbf{y}^T C \leq \mathbf{0}$  has a solution  $\mathbf{y} \in \mathbb{N}^{|P|}$  such that  $\mathbf{y}[i] \geq 1$  for  $i = 1, \dots, |P|$ .
  - $\mathbf{y}$  is a linear combination of *all* place markings that is invariant or decreasing under any transition firing.
- ▶  $N$  is repetitive w.r.t. transition  $t$  iff  $C\mathbf{x} \geq \mathbf{0}$  has a solution  $\mathbf{x} \in \mathbb{N}^{|T|}$  such that  $\mathbf{x}[t] > 0$ .
  - $\mathbf{x}$  is a multiset of transitions, including  $t$  at least once, whose combined effect is zero or increasing.
  - Can always find some initial marking  $\mathbf{m}_0$  from which  $\mathbf{x}$  is realisable.

(... from various places ...)

ICAPS 2009 122 / 140

## Reachability

- ▶ Decidability of the (exact) reachability problem for general Petri nets was open for some time.
  - Algorithm proposed by Sacerdote & Tenney in 1977 incorrect (or gaps in correctness proof).
  - Correct algorithm by Mayr in 1981.
  - Simpler correctness proof (for essentially the same algorithm) by Kosaraju in 1982.
- ▶ Other algorithms have been presented since.
- ▶ All existing algorithms have unbounded complexity.
  - Fun fact: A 2-EXP algorithm was proposed in 1998, but later shown to be incorrect.

(... from various places ...)

ICAPS 2009 123 / 140

## Reachability: Preliminaries

- ▶  $\mathbf{m}$  is *semi-reachable* from  $\mathbf{m}_0$  iff there is a transition sequence  $s = t_{i_1}, \dots, t_{i_n}$  such that  $\mathbf{m} = \mathbf{m}_0 + \mathbf{c}(t_{i_1}) + \dots + \mathbf{c}(t_{i_n})$ .
  - $s$  does not have to be valid (firable) at  $\mathbf{m}_0$ .
  - $\mathbf{m}$  is semi-reachable from  $\mathbf{m}_0$  iff  $C\mathbf{n} = (\mathbf{m} - \mathbf{m}_0)$  has a solution  $\mathbf{n} \in \mathbb{N}^{|T|}$ .
- ▶ If  $\mathbf{m}$  is semi-reachable from  $\mathbf{m}_0$ , then  $\mathbf{m} + \mathbf{a}$  is reachable from  $\mathbf{m}_0 + \mathbf{a}$  for some sufficiently large  $\mathbf{a} \geq \mathbf{0}$ .

(... from various places ...)

ICAPS 2009 124 / 140

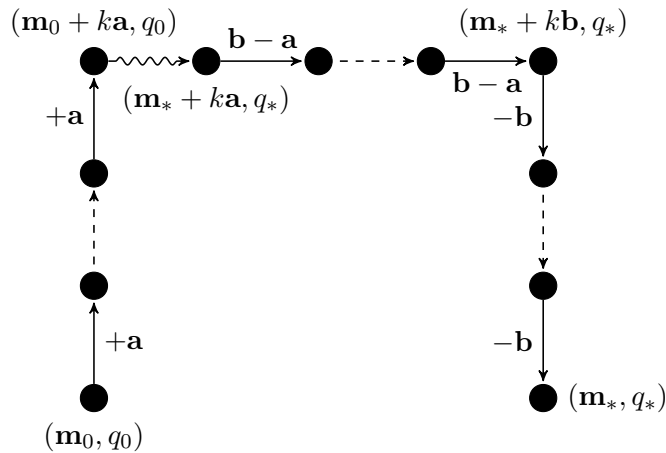
- ▶ A *controlled net* is a pair of a marked net  $(N = \langle P, T, F \rangle, \mathbf{m}_0)$  and an NFA  $(A, q_0)$  over alphabet  $T$ .
  - $A$  defines a (regular) subset of (not necessarily firable) transition sequences.
  - Define reachability/coverability/boundedness for  $(N, \mathbf{m}_0)$  w.r.t.  $A$  in the obvious way.
  - The coverability tree construction is easily modified to consider only sequences accepted by  $A$ .
- ▶ The *reverse* of  $N$ ,  $N_{\text{Rev}}$  (w.r.t.  $A$ ) is obtained by reversing the flow relation (and arcs in  $A$ ).
  - $C(N_{\text{Rev}}) = -C(N)$ .

(... from various places ...)

## Reachability: A Sufficient Condition

- ▶ In  $(N, \mathbf{m}_0)$  w.r.t.  $(A, q_0)$ , if
  - (a)  $(\mathbf{m}_*, q_*)$  is semi-reachable from  $(\mathbf{m}_0, q_0)$ ,
  - (b)  $(\mathbf{m}_0 + \mathbf{a}, q_0)$  is reachable from  $(\mathbf{m}_0, q_0)$ , for  $\mathbf{a} \geq 1$ ,
  - (c)  $(\mathbf{m}_* + \mathbf{b}, q_*)$  is reachable from  $(\mathbf{m}_*, q_*)$  in  $N_{\text{Rev}}$  w.r.t.  $A$ , for  $\mathbf{b} \geq 1$ ,
  - (d)  $(\mathbf{b} - \mathbf{a}, q_*)$  is semi-reachable from  $(\mathbf{0}, q_*)$ , then  $(\mathbf{m}_*, q_*)$  is reachable from  $(\mathbf{m}_0, q_0)$ .
- ▶ The conditions above are effectively checkable:
  - (b) & (c) by coverability tree construction,
  - (a) & (d) through the state equation.

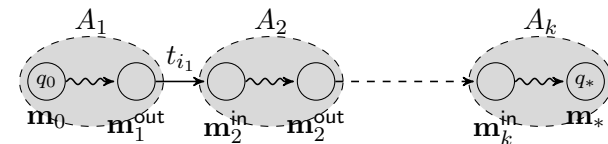
(... from various places ...)



(... from various places ...)

## Reachability: The Mayr/Kosaraju Algorithm

Consider a controlled net  $(N, A)$  of the form,



with constraints  $\mathbf{m}_i^{\text{in/out}}[j] = x_{i,j}^{\text{i/o}}$  or  $\mathbf{m}_i^{\text{in/out}}[j] \geq y_{i,j}^{\text{i/o}} \geq 0$ .

- ▶ If the sufficient reachability condition holds for each  $(\mathbf{m}_i^{\text{in}}, q_i^{\text{in}})$  and  $(\mathbf{m}_i^{\text{out}}, q_i^{\text{out}})$  w.r.t.  $A_i$ , then  $(\mathbf{m}_*, q_*)$  is reachable from  $(\mathbf{m}_0, q_0)$ .
- ▶ Let  $\Delta(A_i) = \{\mathbf{m} \mid \mathbf{m} = C\mathbf{n}(s), s \in L(A_i)\}$ .
- ▶ Let  $\Gamma = \{\mathbf{m}_i^{\text{in}}, \mathbf{m}_i^{\text{out}}, \mathbf{n}_i \mid \mathbf{m}_{i+1}^{\text{in}} - \mathbf{m}_i^{\text{out}} = \mathbf{c}(t_{i,i}), \mathbf{m}_i^{\text{out}} - \mathbf{m}_i^{\text{in}} \in \Delta(A_i), \text{ and constraints hold}\}$ .
- ▶ If  $(\mathbf{m}_0, q_0) [s] (\mathbf{m}_*, q_*)$ ,  $s$  defines an element in  $\Gamma$ .

(... from various places ...)




- ▶  $\Gamma$  is a *semi-linear set*: consistency (non-emptiness) is decidable via Pressburger arithmetic.
- ▶ If  $\Gamma$  is consistent, but the sufficient condition does not hold in some  $A_i$ , then  $A_i$  can be replaced by a new “chain” of controllers,  $A_i^1, \dots, A_i^{l_i}$ , each of which is “simpler”:
  - more equality constraints ( $\mathbf{m}_{i^l}^{\text{in/out}} = x_{i^l, j}^{\text{i/o}}$ ), or
  - same equality constraints and smaller automaton.
- ▶ There can be several possible replacements (non-deterministic choice).
- ▶ If  $(\mathbf{m}_*, q_*)$  is not reachable from  $(\mathbf{m}_0, q_0)$ , every choice (branch) eventually leads to an inconsistent system.

## FSMs, Marked Graphs & Free Choice Nets

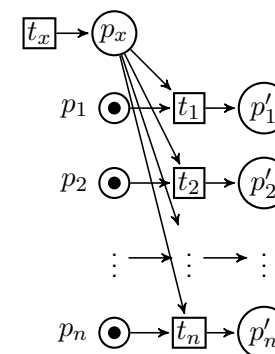
- ▶ An ordinary Petri net with  $|\bullet t| = |t\bullet| = 1$  for each transition  $t$  is a *P-graph*, or *finite state machine*.
  - Every P-graph is structurally bounded (# of tokens is constant).
- ▶ An ordinary Petri net with  $|\bullet p| = |p\bullet| = 1$  for each place  $p$  is a *T-graph*, or *marked graph*.
  - Several properties of marked graphs (e.g., liveness, boundedness, 1-safety) are decidable in polynomial time.
- ▶ An ordinary Petri net such that  $|p\bullet| \leq 1$  or  $\bullet(p\bullet) = \{p\}$  for each place  $p$ , is a *free choice* net.

## Characterisation by Derivation Rules

- ▶ Initial net: 
- ▶ Rule #1: Add a new place  $p'$  with  $\mathbf{r}(p') = \sum_{p \in P} \lambda_p \mathbf{r}(p)$  and  $|p'\bullet| = 1$ .
- ▶ Rule #2: Replace place  $p$  with a connected P-graph  $N'$ , and connect each input and output of  $p$  to at least one place in  $N'$ .
  - Must have  $|\bullet p| > 1$  and  $|p\bullet| > 1$ , except for initial net.
  - Every place  $p' \in N'$  must appear on a path in the resulting net that enters and leaves  $N'$ .
- ▶ **Theorem:** The class of nets obtained by applying the above rules to the initial net is exactly the class of structurally live and structurally bounded free choice nets.

## Acyclic Nets

- ▶ For an acyclic net, every solution to  $C\mathbf{n} = (\mathbf{m} - \mathbf{m}_0)$  is realisable.
- ▶ A minimum cost firing sequence can be found by ILP (and lower-bounded by LP).
- ▶ Removing transition input arcs is a relaxation.
- ▶ We have a new heuristic!



Minimum cost:  $2n$ .  
 LP relaxation:  $2n$ ?  
 $h^+$ :  $n + 1$ .

## Summary & Conclusions

- ▶ 1-Safe Petri nets: Intuitive, graphical modelling formalism, closely related to planning.
- ▶ Unfolding: Search that combines partial-order planning with state-space heuristics.
- ▶ Petri net theory often uses different tools than planning:
  - Algebraic methods (based on the state equation).
  - Characterisation and study of classes of nets with special structure.
- ▶ Planning also has tools applicable to Petri nets:
  - Domain-independent search heuristics.
  - What else?

(... from various places ...)

ICAPS 2009 133 / 140

## The Many Things We Haven't Talked About



- ▶ Extensions of basic Place-Transition nets:
  - Read arcs, reset arcs and inhibitor arcs.
  - Colored Petri nets, timed nets, stochastic nets, etc.
- ▶ Other properties of Petri nets (and related decision problems):
  - Model checking (tense logics, process calculi).
  - Language (trace) properties.
- ▶ Heaps more results concerning different Petri net subclasses.

(... from various places ...)


ICAPS 2009 134 / 140

## Bibliography I

### Introductions & Survey Papers

-  Murata, T. 1989.  
Petri Nets: Properties, Analysis and Applications  
*In Proceedings of the IEEE 77(4):541–580.*
-  Esparza, J. 1998.  
Decidability and complexity of Petri net problems – an introduction.  
*In Lectures on Petri Nets I: Basic Models. LNCS 1491:374–428.*





### Unfolding

-  McMillan, K. L. 1992.  
Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits.  
*In Computer Aided Verification, 4th International Workshop (CAV'92). LNCS 663:164–177.*

(... from various places ...)

ICAPS 2009 135 / 140




## Bibliography II

-  McMillan, K. L. 1995.  
A technique of state space search based on unfolding.  
*Formal Methods in System Design, 6:45–65.*
-  Vogler, W., Semenov, A. L., and Yakovlev, A. 1998.  
Unfolding and finite prefix for nets with read arcs.  
*In CONCUR '98: Concurrency Theory, 9th International Conference. LNCS 1466:501–516.*
-  Esparza, J., Römer, S. and Vogler, W. 2002.  
An Improvement of McMillan's Unfolding Algorithm.  
*Formal Methods in System Design 20(3):285–310.*
-  Victor Khomenko, Maciej Koutny, and Walter Vogler. 2002.  
Canonical prefixes of Petri net unfoldings.  
*In Computer Aided Verification, 14th International Workshop (CAV'02). LNCS 2404:164–177.*

(... from various places ...)

ICAPS 2009 136 / 140




## Bibliography III

-  Hickmott, S., Rintanen, J., Thiébaux, S. and White, L. 2007.  
Planning via Petri net unfolding.  
*In Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 1904–1911.
-  Esparza, J., Kanade, P. and Schwoon, S. 2008.  
A negative result on depth-first net unfoldings.  
*International Journal on Software Tools for Technology Transfer* 10(2):161–166.
-  Bonet, B., Haslum, P., Hickmott, S. and Thiébaux, S. 2008.  
Directed Unfolding of Petri Nets.  
*Transactions on Petri Nets and Other Models of Concurrency* 1:172-198 (LNCS 5100).

(... from various places ...)

ICAPS 2009 137 / 140



## Bibliography IV

-  Javier Esparza and Keijo Heljanko.  
*Unfoldings – A Partial-Order Approach to Model Checking*.  
EATCS Monographs in Theoretical Computer Science.  
Springer-Verlag, March 2008.
- Reachability in General Petri Nets**
-  Mayr, E. W. 1984.  
An Algorithm for the General Petri Net Reachability Problem.  
*SIAM Journal on Computing* 13(3):441–460.  
(*This is the journal version; the result was first published in ACM STOC 1981.*)
-  Kosaraju, S. R. 1982.  
Decidability of Reachability in Vector Addition Systems.  
*In Proc. 14th ACM Symposium on the Theory of Computing* 267–281.


(... from various places ...)

ICAPS 2009 138 / 140

## Bibliography V

-  Kostin, A. 2006.  
A Reachability Algorithm for General Petri Nets Based on  
Transition Invariants.  
*In Mathematical Foundations of Computer Science. LNCS* 4162:608–621.
-  Jancar, P. 2008.  
Bouziane's transformation of the Petri net reachability problem  
and incorrectness of the related algorithm.  
*Information and Computation* 206(11):1259-1263.

### Nets with Special Structure


-  Esparza, J. 1990.  
Synthesis Rules for Petri Nets, and How they Lead to New  
Results.  
*In CONCUR'90, Theories of Concurrency: Unification and  
Extension. LNCS* 458:182–189.

(... from various places ...)

ICAPS 2009 139 / 140

## Bibliography VI

### Extended Petri Net Models

-  Jensen, K. 1989.  
Coloured Petri nets: A high level language for system design and  
analysis.  
*In Applications and Theory of Petri Nets*, pages 342–416.

(... from various places ...)

ICAPS 2009 140 / 140