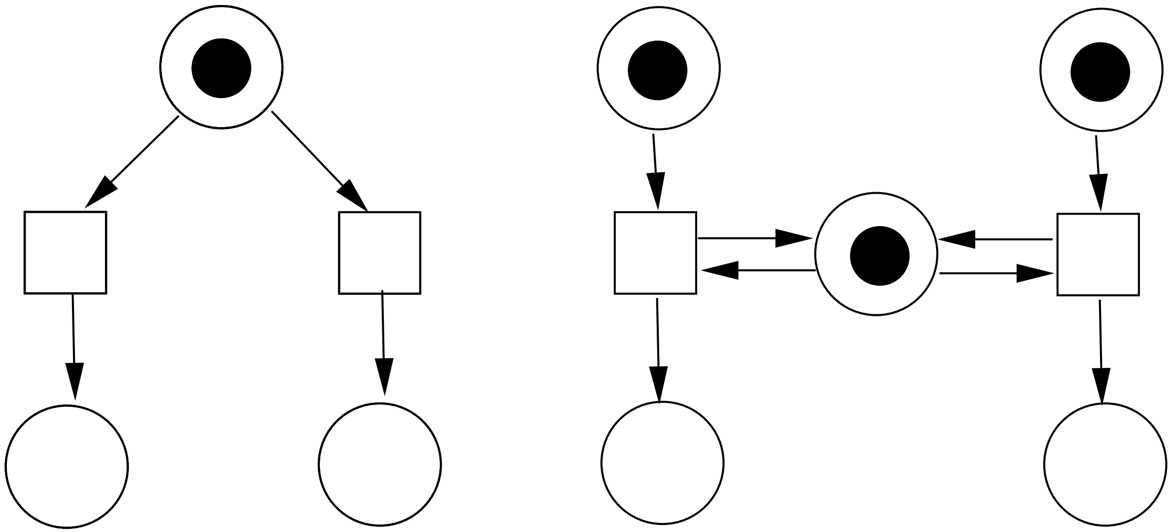
# Unfoldings for Contetxtual Petri Nets

Paolo Baldan (Padova), Andrea Corradini (Pisa),

Barbara König (Duisburg-Essen), Stefan Schwoon (Cachan),

# Why Petri nets?

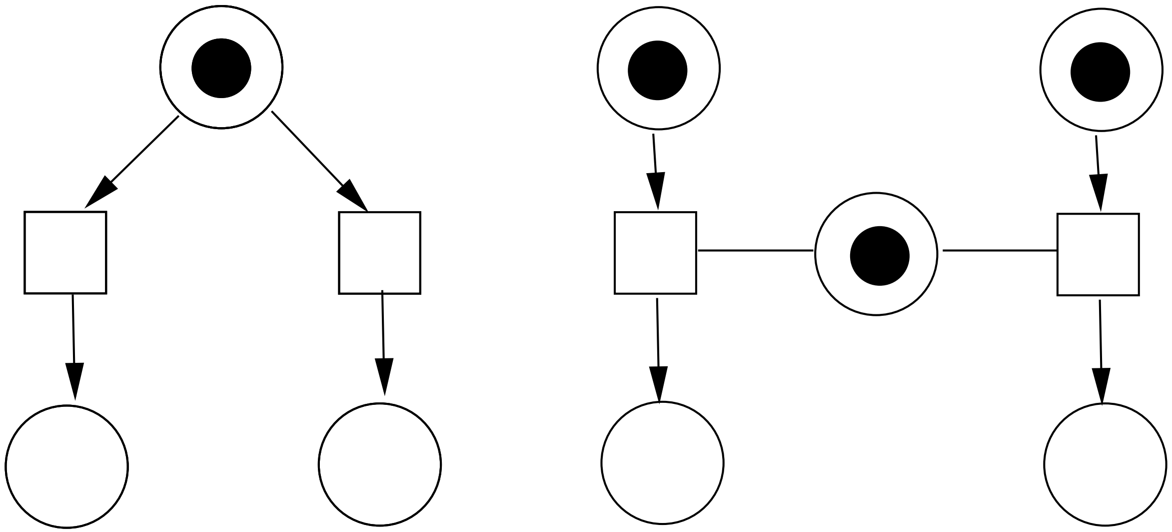
Model for distributed, concurrent system:



Expresses independence, conflict, causality, . . .

# Why contextual Petri nets?

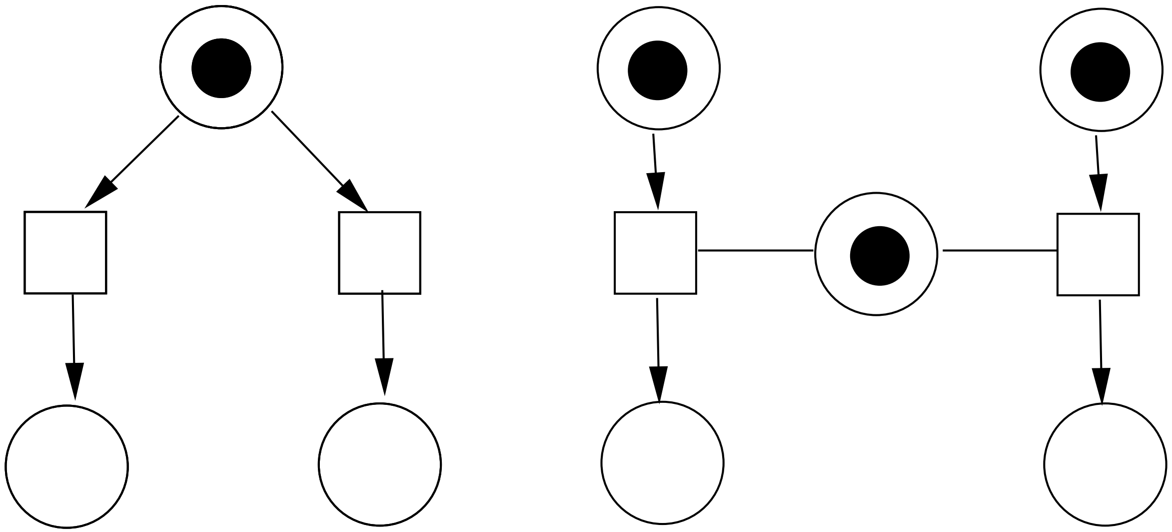
Explicit modelling of “read/test” actions (arcs w/o arrows):



Intuition: The read arc does not consume or touch the token, it merely verifies its presence. For any transition $t$, we distinguish its preset ${}^\bullet t$, its context $\underline{t}$, and its postset $t^\bullet$.

# Why contextual Petri nets?

Explicit modelling of “read/test” actions (arcs w/o arrows):

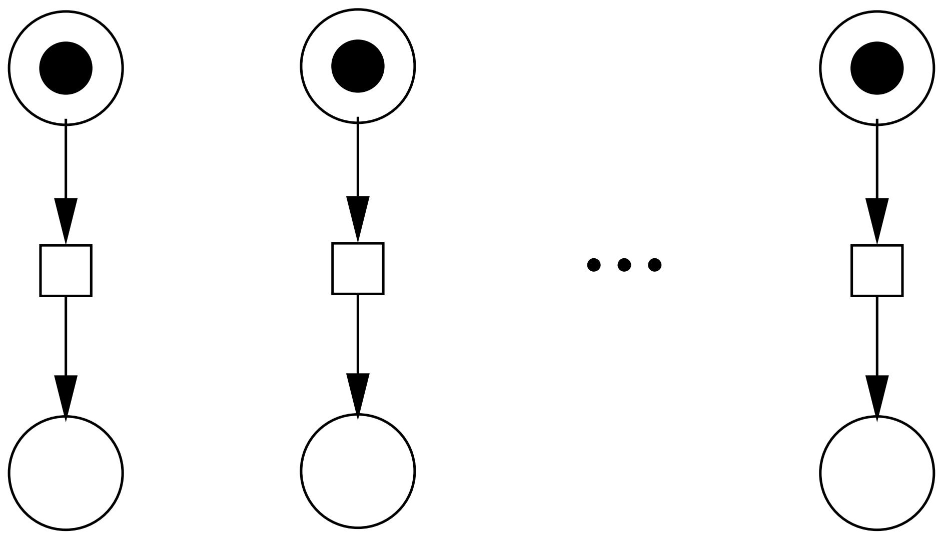


Different concurrent semantics (no difference for interleaving semantics).

Same set of reachable markings. Here, we are interested in their unfoldings.

# Reachability analysis for Petri nets

For bounded nets, the reachability graph is finite.



However, it explodes in the presence of concurrency.

# Unfoldings

Data structure for representing the reachable markings, exploits concurrency inherent in the Petri net model.

Size between that of Petri net and that of reachability graph; once unfolding is computed, reachability queries become easier.

Unfoldings for "normal" Petri nets established by McMillan (1992), a lot of other work since then, see, e.g., the book by Esparza, Heljanko for a survey.

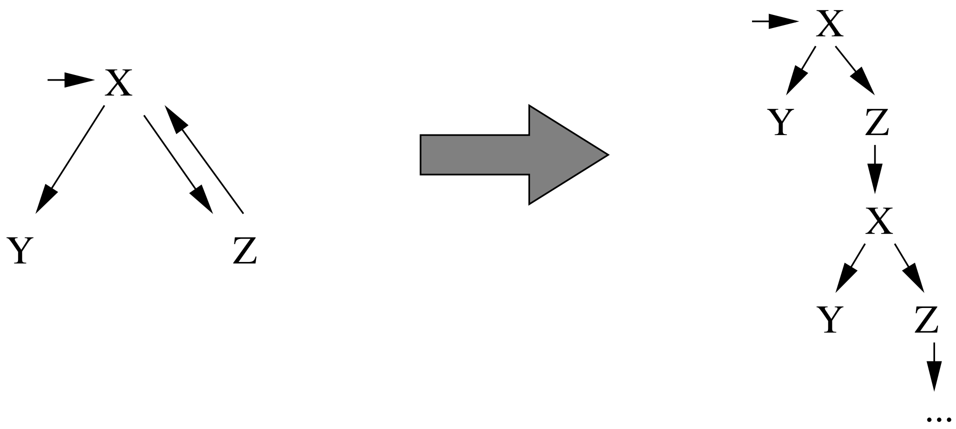
Unfoldings for contextual nets:

> for read-persistent subclass: Vogler, Semenov, Yakovlev (1998) and Baldan, Corradini, Montanari (1998)
>
> for general nets, but non-constructive: Winkowski (2002)

Will explain ideas first for "normal" Petri nets (without read arcs).

# Unfoldings for finite automata

The unfolding of a finite automaton is its computation tree:




Principles:

The unfolding of a finite automaton is an *acyclic*, infinite automaton.

The unfolding has the same behaviours and the same reachable states.

Construction: Start with initial state; for every state in the unfolding and each outgoing transition, add a *fresh copy* of the target.

# Unfoldings for (normal) Petri nets

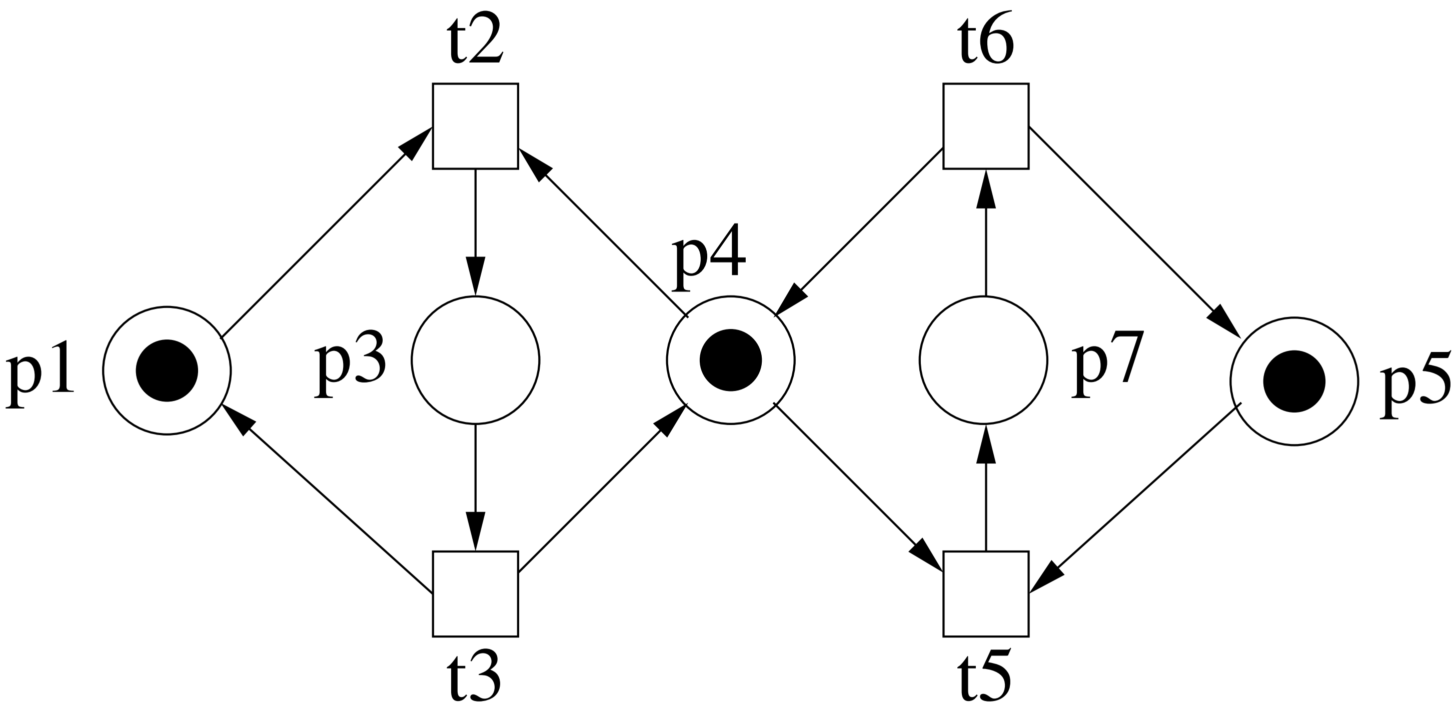
Principles:

The unfolding of a Petri net is an *acyclic*, infinite Petri net.

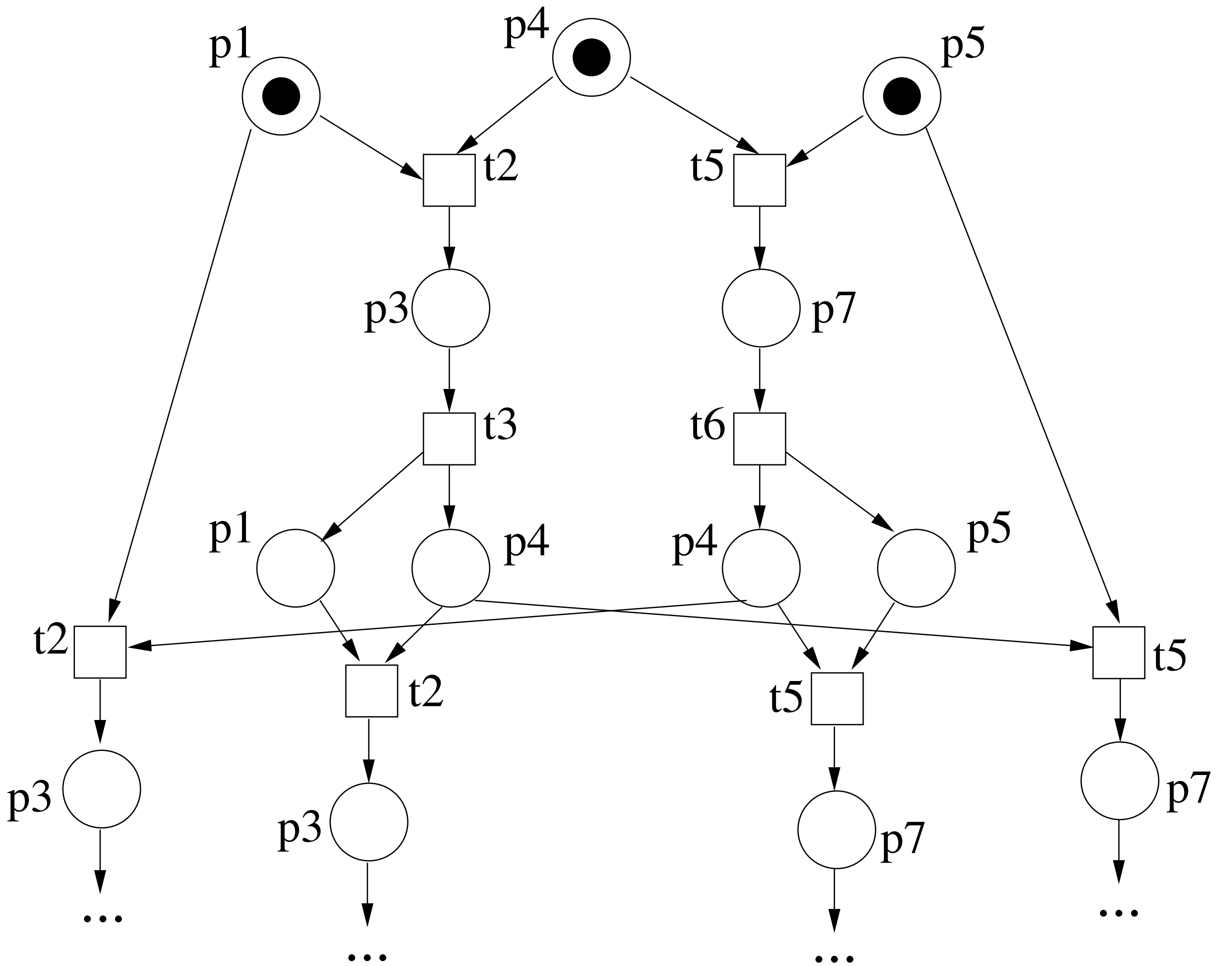
The unfolding has the same behaviours and the same reachable states.

Construction: Start with initially marked places; for every coverable marking that enables a transition, add that transition with *fresh copies* of the output places.

# Example: Petri net. . .

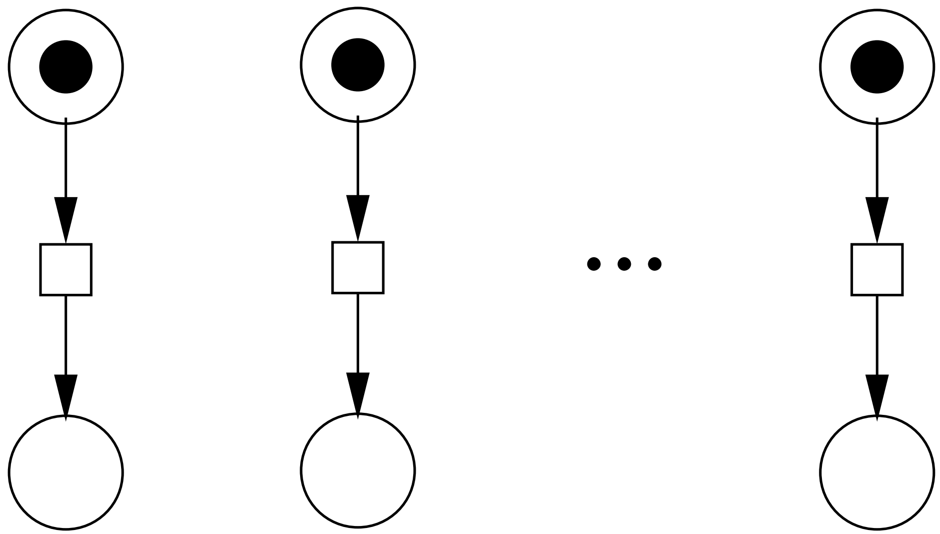


# . . . and its unfolding



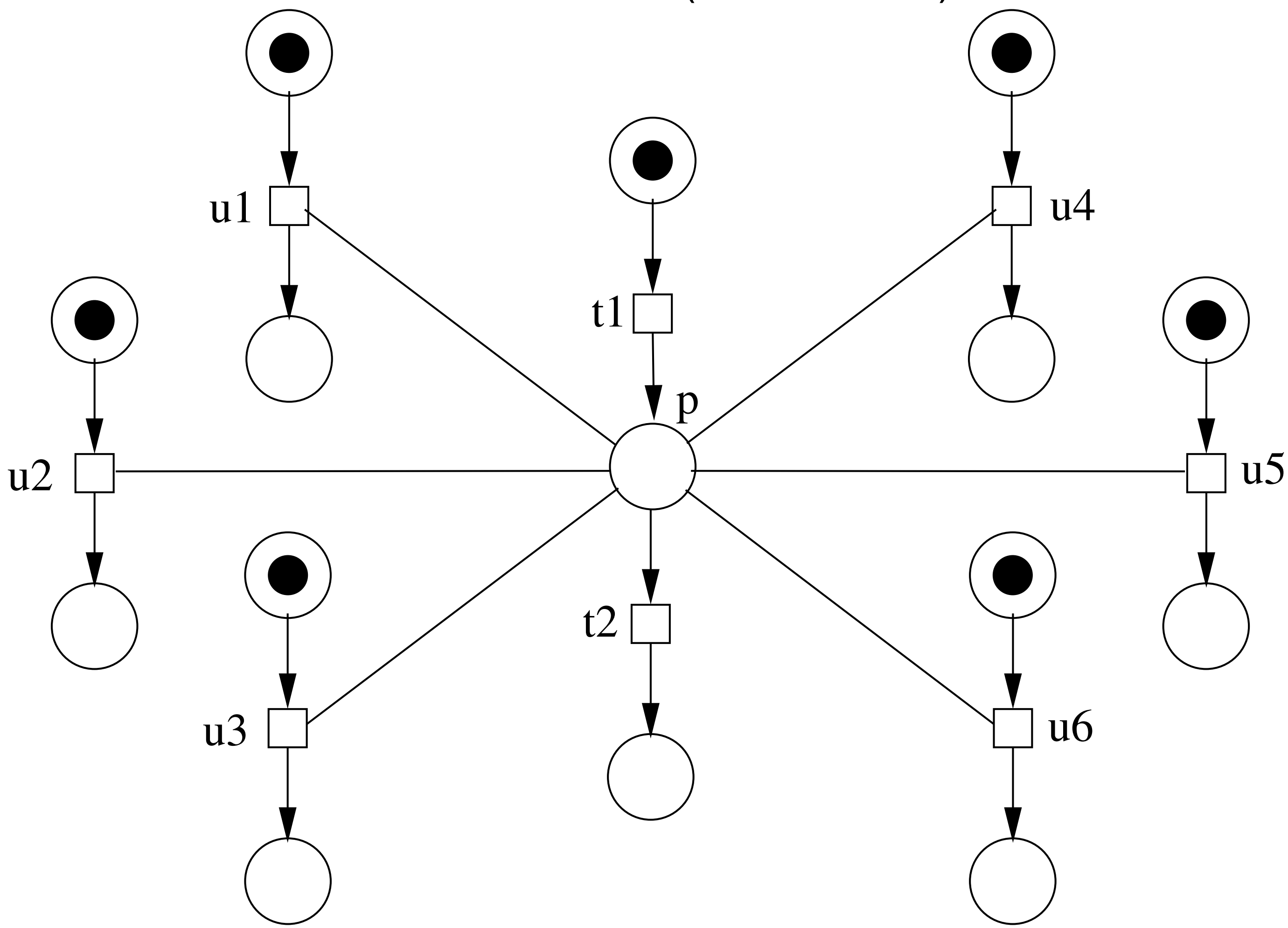
# Unfoldings exploit concurrency

The net shown below and its unfolding are identical.



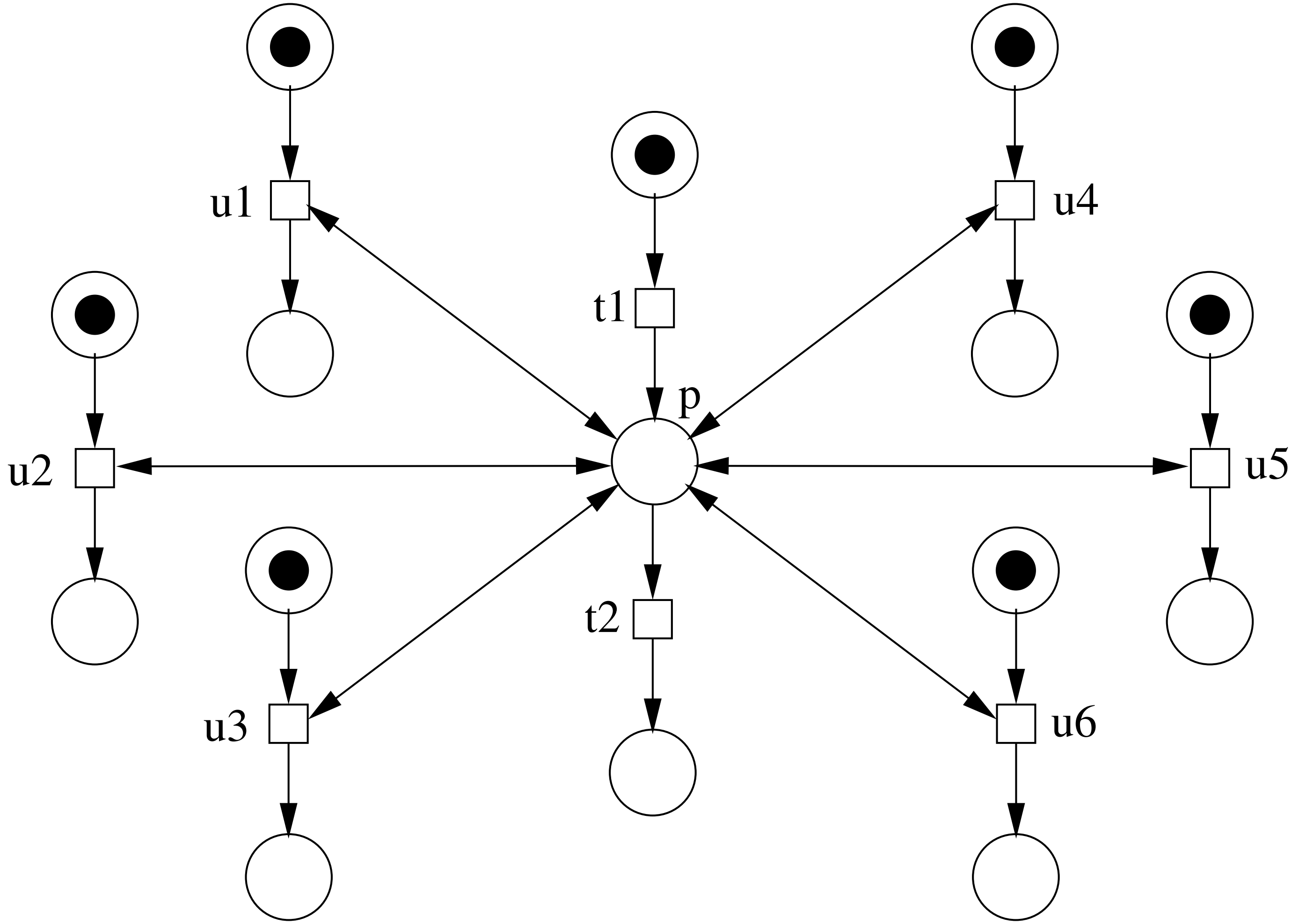
# Unfoldings of contextual nets

Consider the contextual net shown below (six readers):



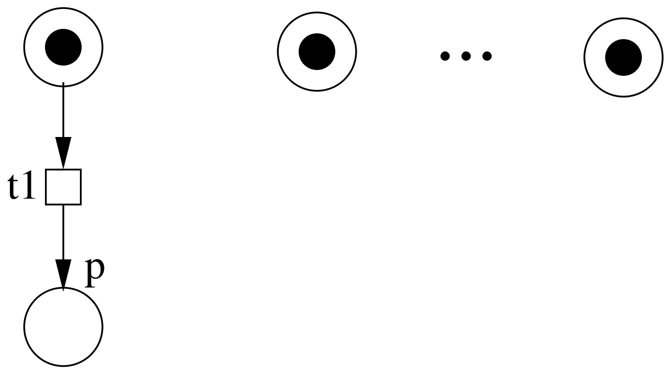
# Unfoldings of contextual nets: Naïve approach

Why not replace read arcs by double arrows and unfold normally?



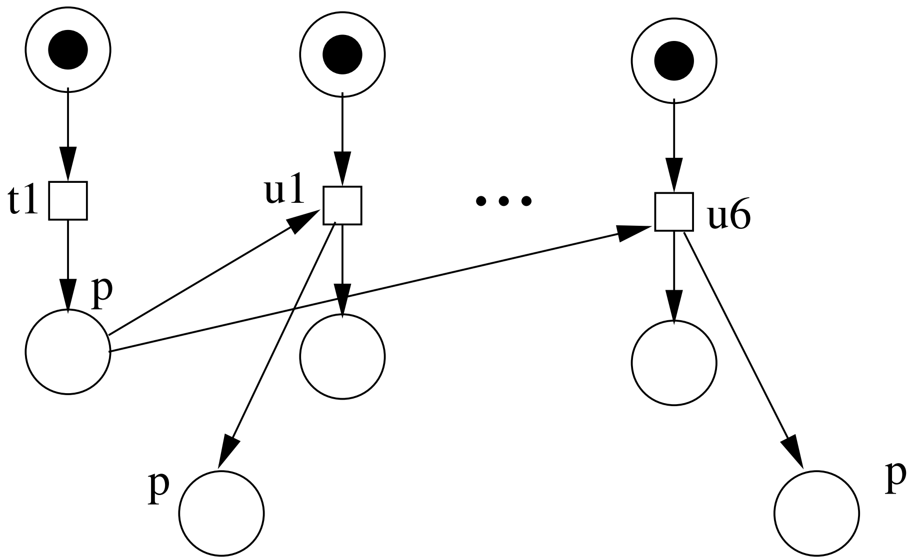
# Unfoldings of contextual nets: Naïve approach

Here's why: Initial addition of $t_1$, . . .



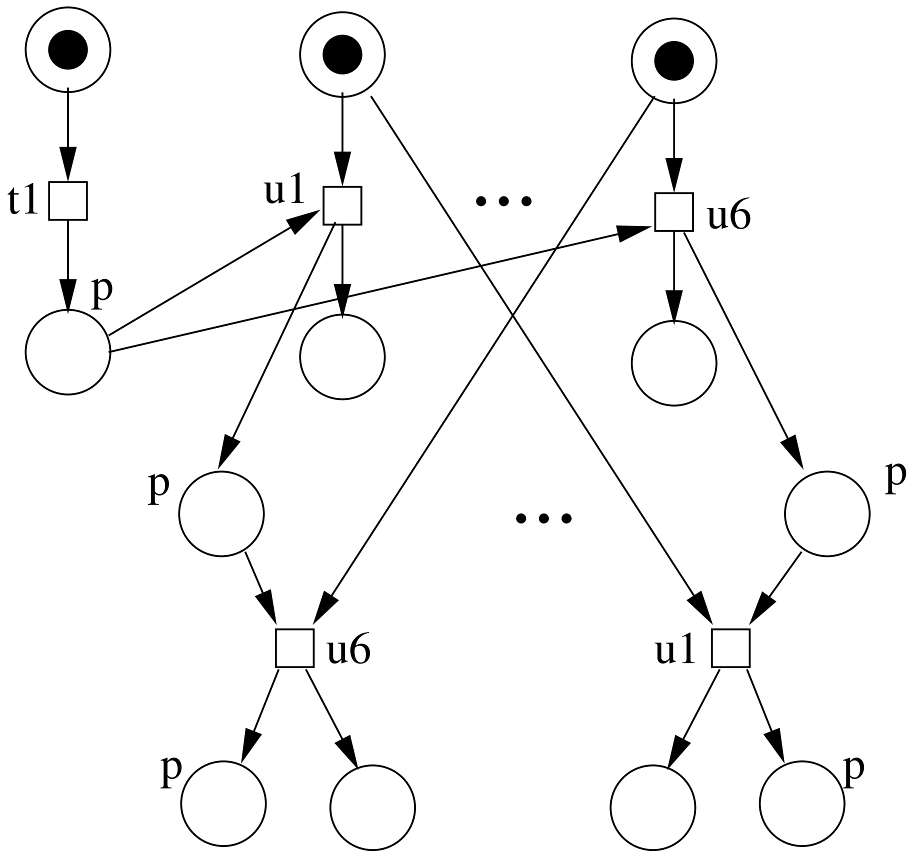
# Unfoldings of contextual nets: Naïve approach

. . . followed by copies of $u_1, \ldots, u_6$, generating "second-generation" copies of $p$.



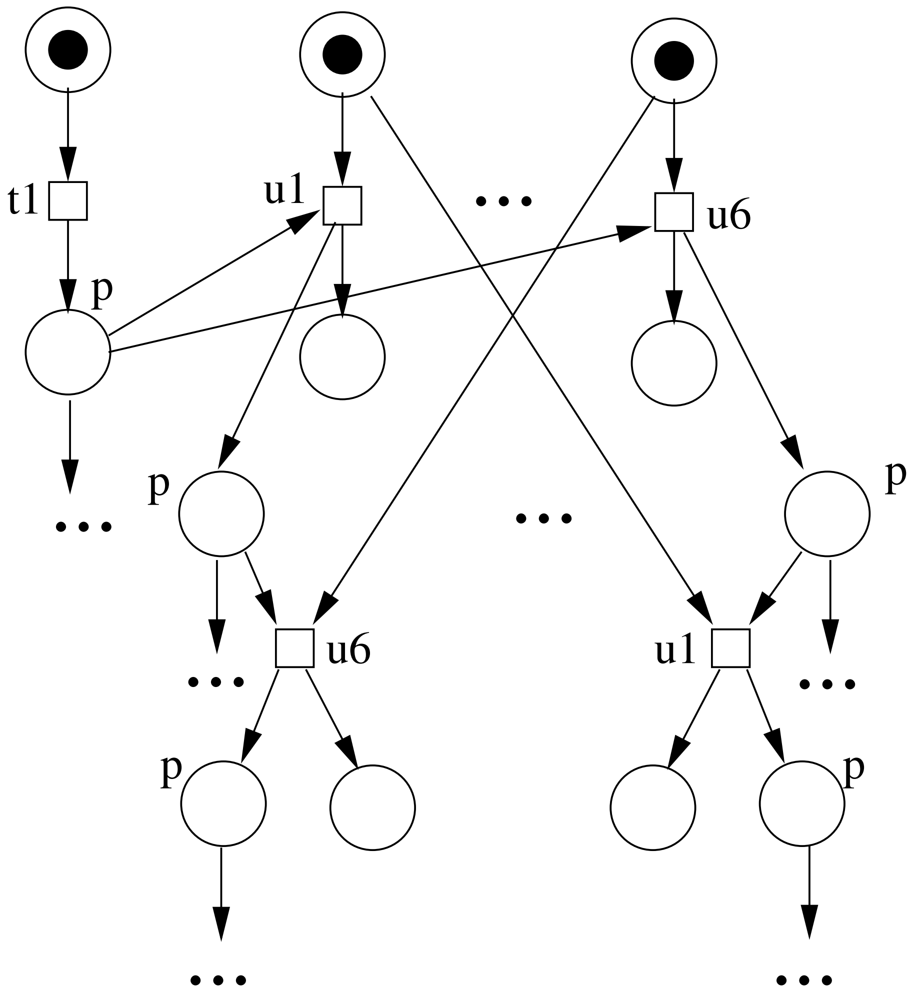
# Unfoldings of contextual nets: Naïve approach

Second layer of $u_i$ transitions using the new copies of $p$, generating more of them etc.



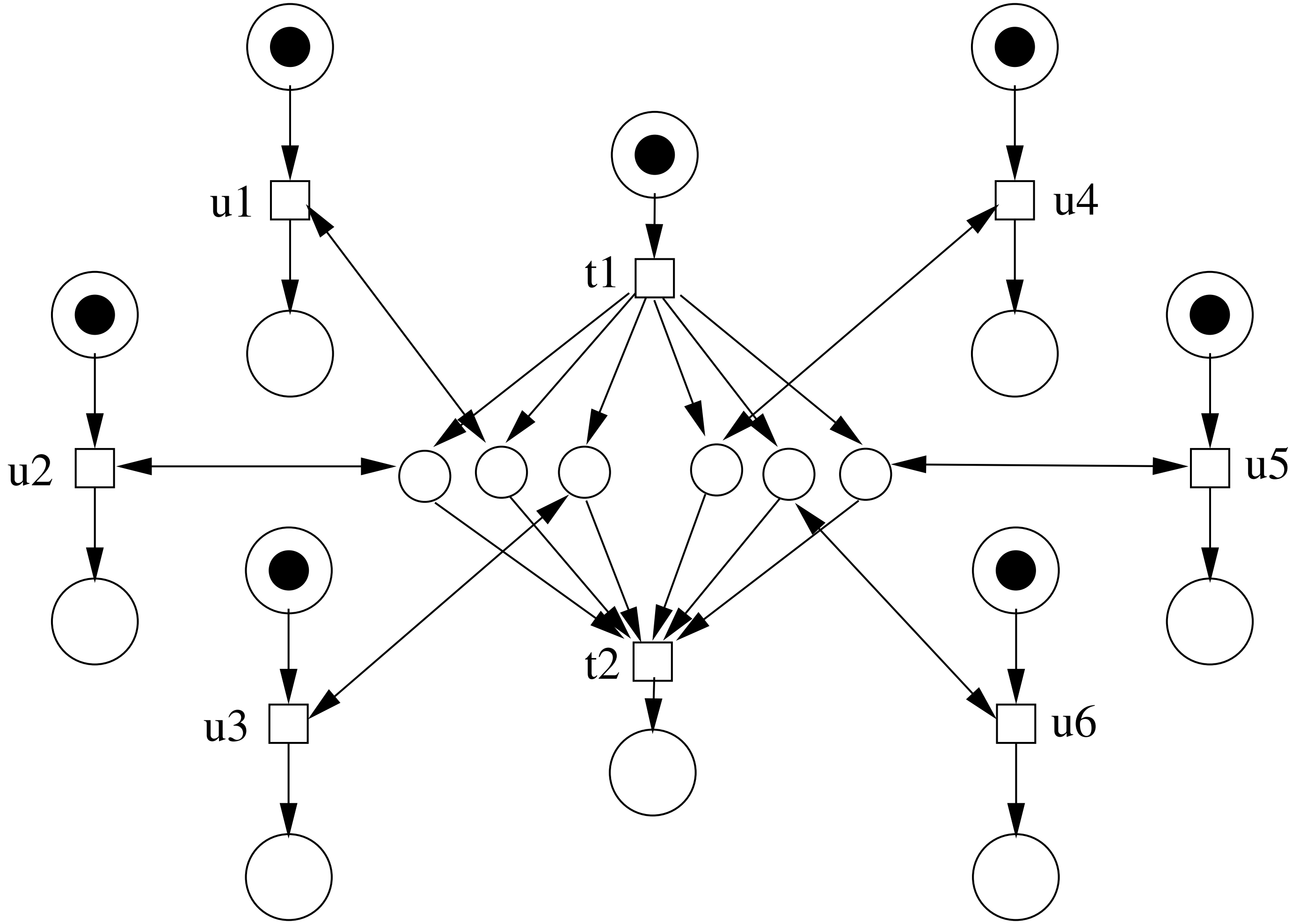
# Unfoldings of contextual nets: Naïve approach

Altogether, one event for every permutation of $1 \ldots 6$, i.e. $6!$



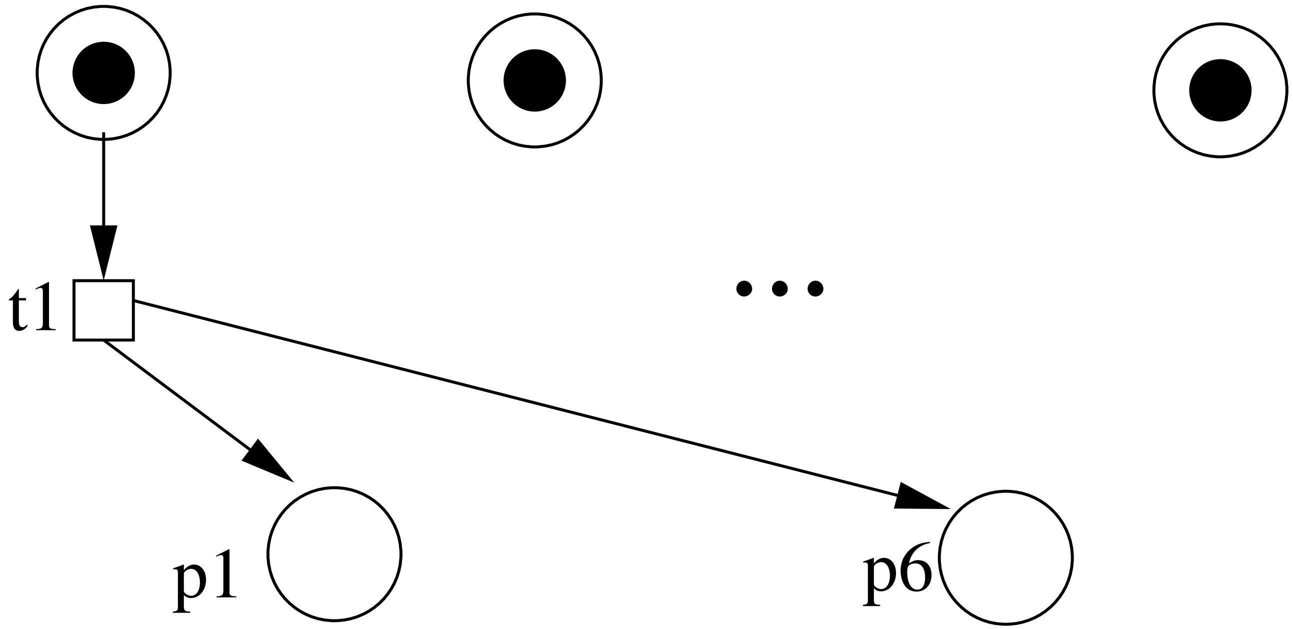
# Unfoldings of contextual nets: PR-approach

Less naïve: Replace *p* by six copies, one for each reader.



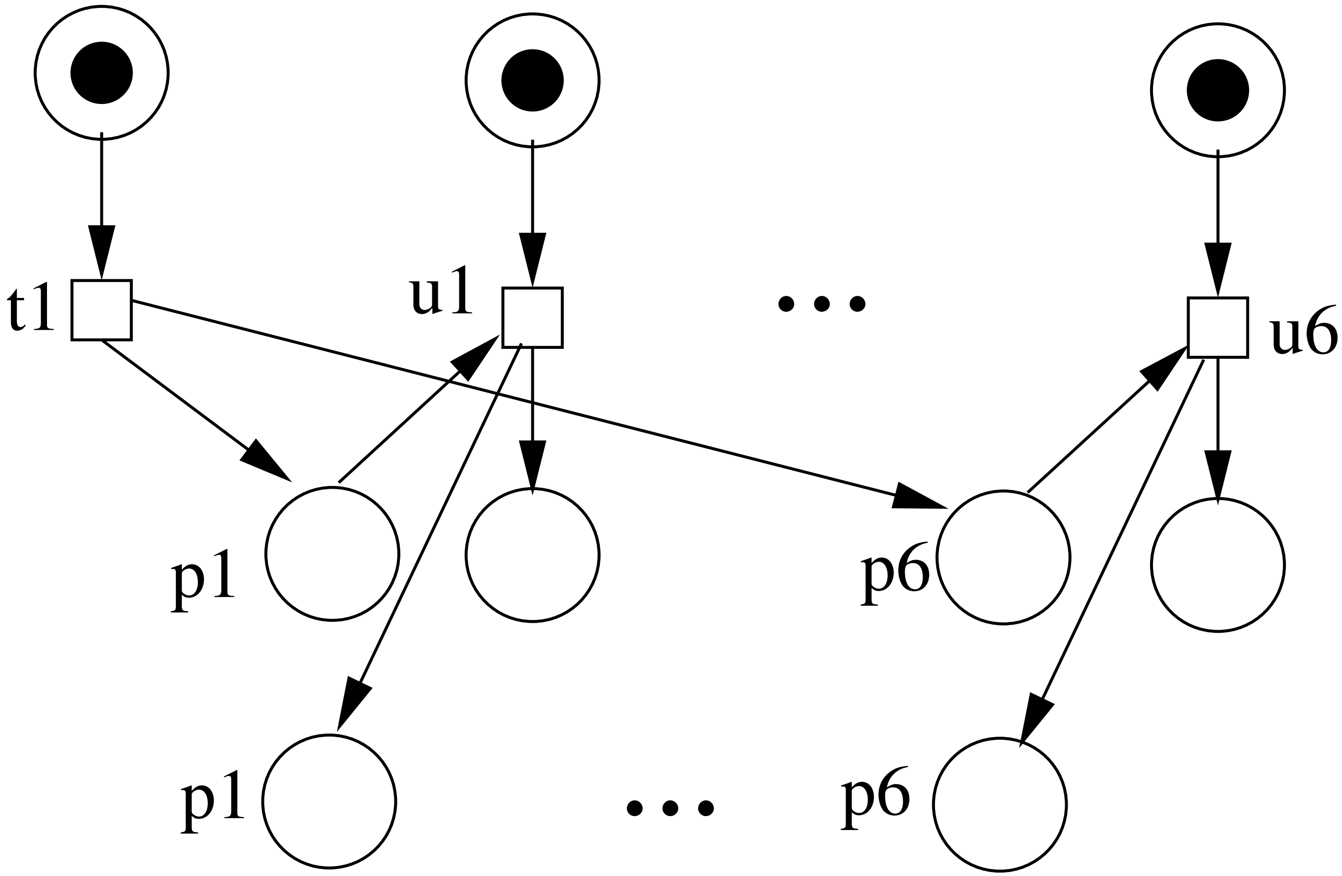
# Unfoldings of contextual nets: PR-approach

Unfolding starts with one copy of $t_1$, . . .



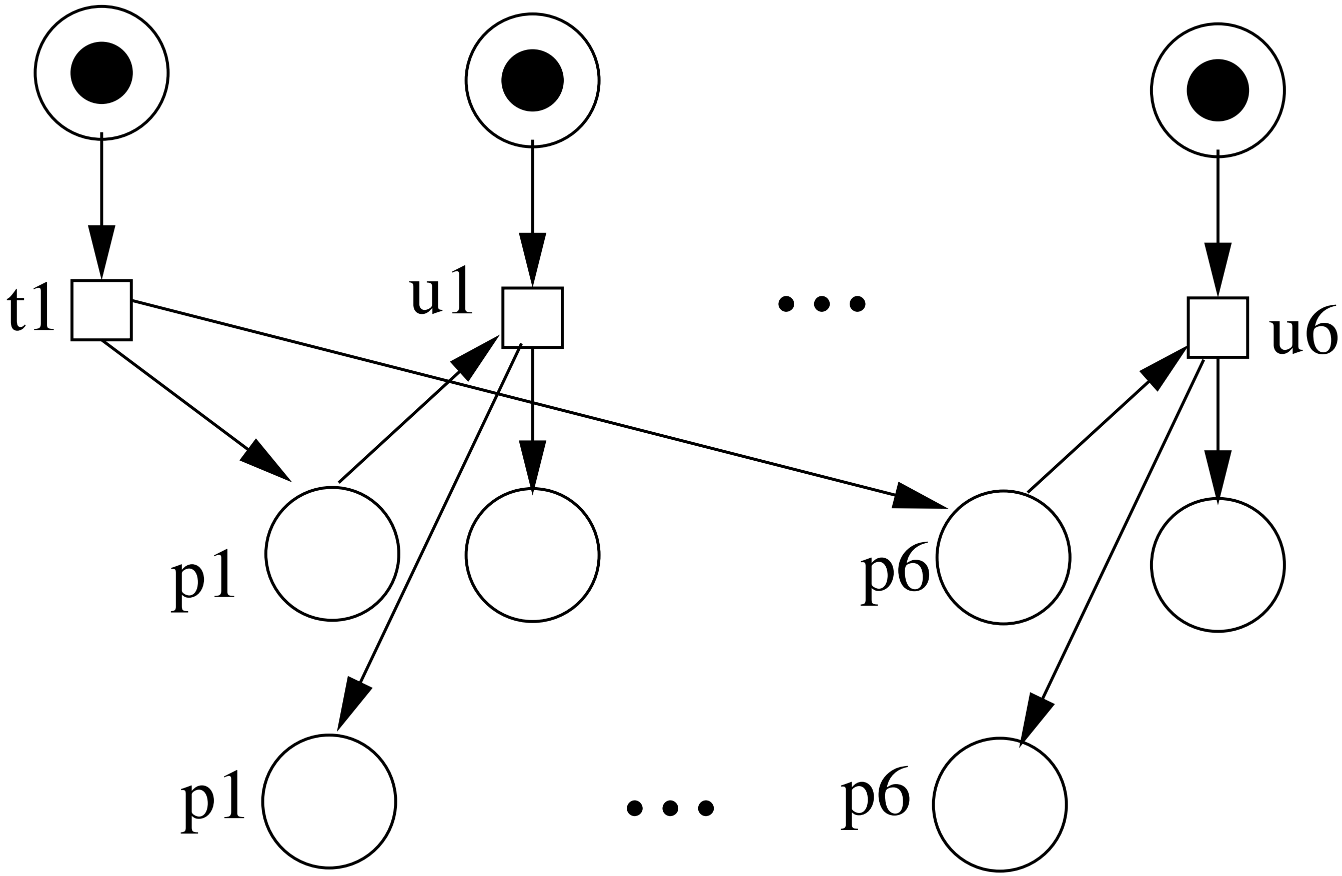
# Unfoldings of contextual nets: PR-approach

...then just one copy each of $u_1, \ldots, u_6$!



# Unfoldings of contextual nets: PR-approach

However, we will still have $2^6$ copies of $t_2$...



# Unfoldings of contextual nets: Our approach

Neither encoding (naïve, PR) of contextual nets into normal Petri nets yields satisfying results.

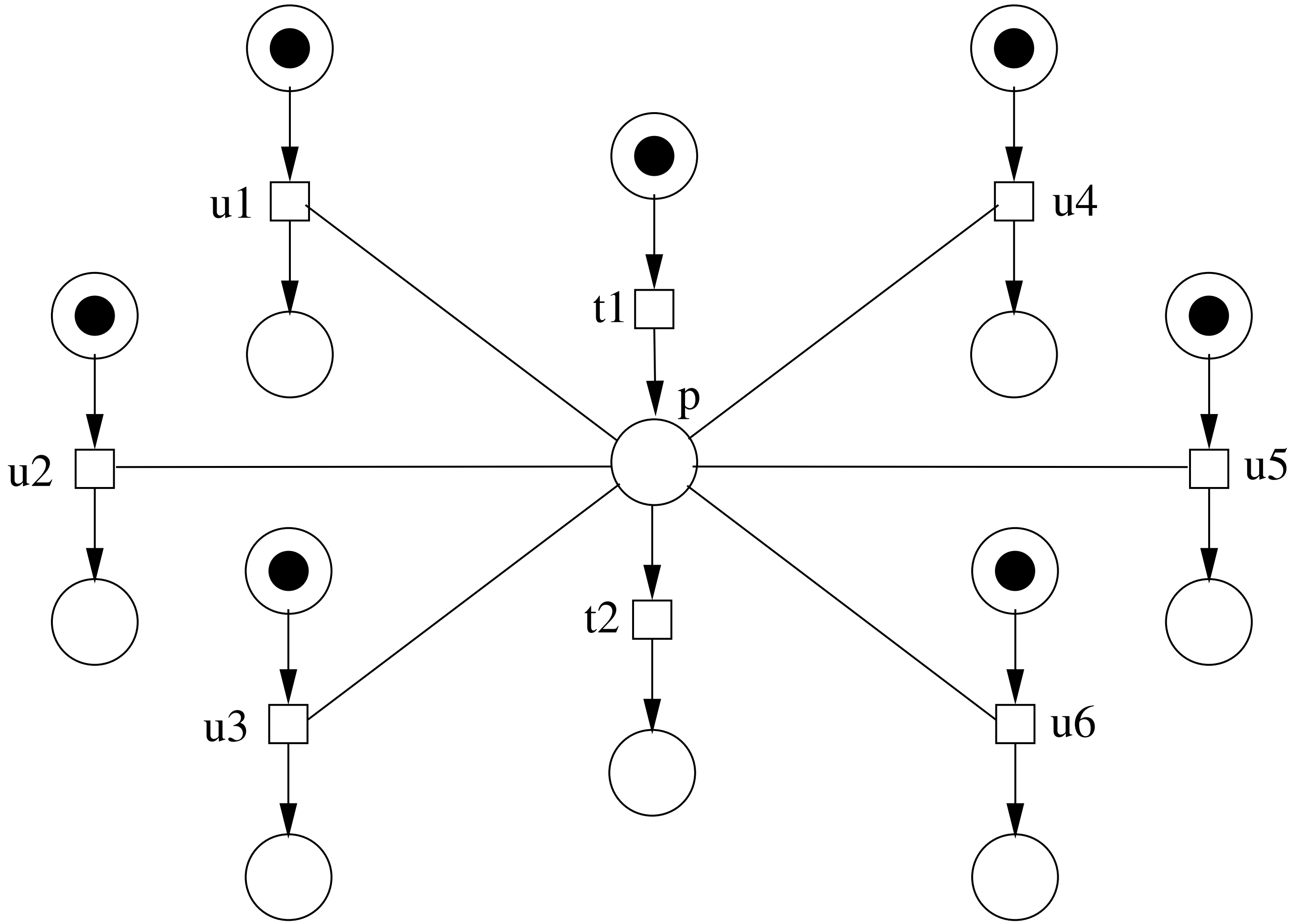
We propose a new, direct unfolding procedure for contextual Petri nets that avoids blowup in the presence of concurrent readers.

Principles:

- The unfolding of a contextual net is an *acyclic*, infinite contextual net.
- The unfolding has the same behaviours and the same reachable states.
- Construction: Start with initially marked places; for every coverable marking that enables a transition, add that transition with *fresh copies* of the output places but only read arcs to its context.

# Example (six readers)

The contextual net shown below is identical to its unfolding.



# Algorithmic problems

Decide (efficiently) whether a set of places is coverable.

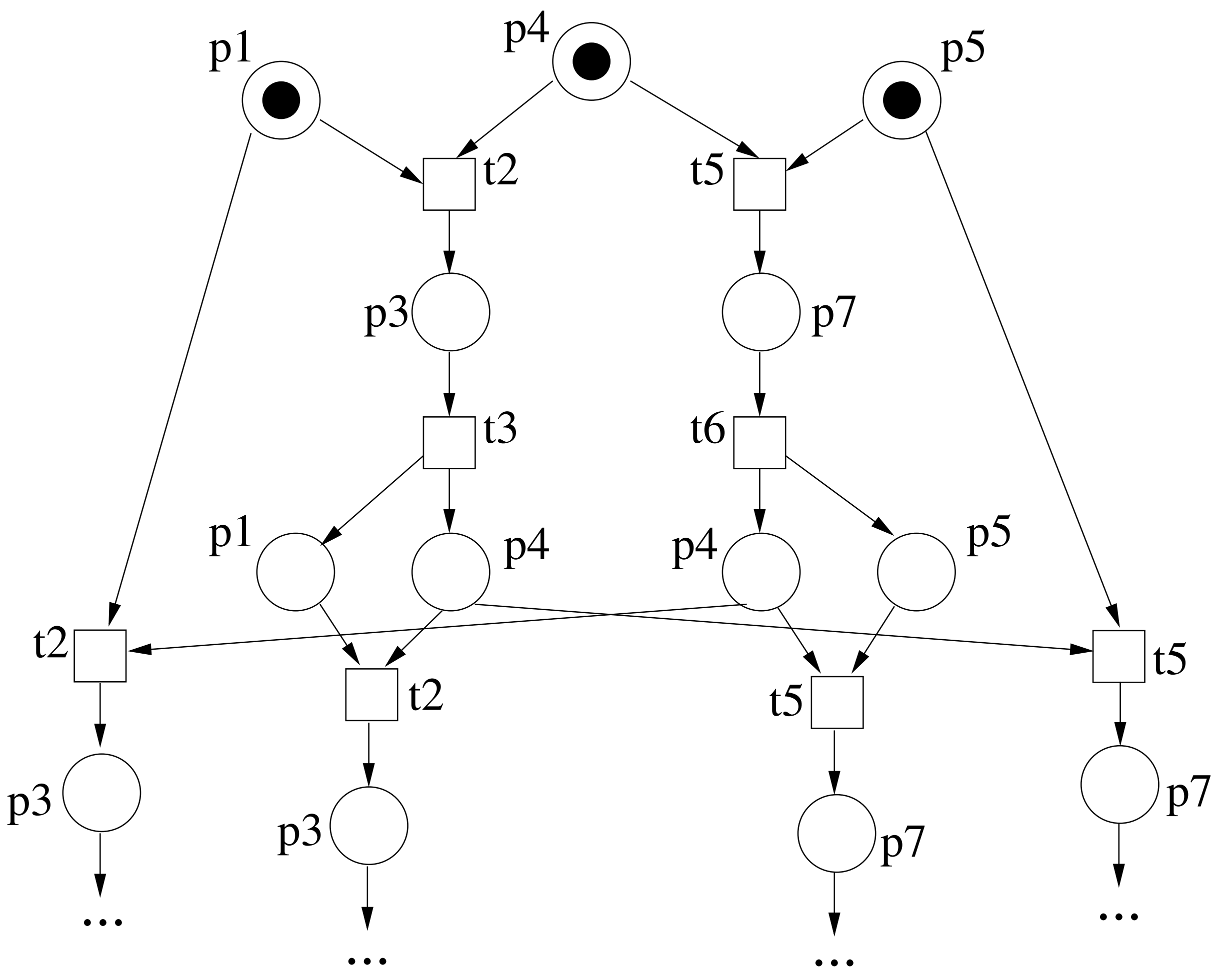
$\longrightarrow$ decision required whenever the unfolding is extended

How to compute a complete finite prefix of the unfolding?

$\longrightarrow$ complete $\widehat{=}$ contains all reachable markings

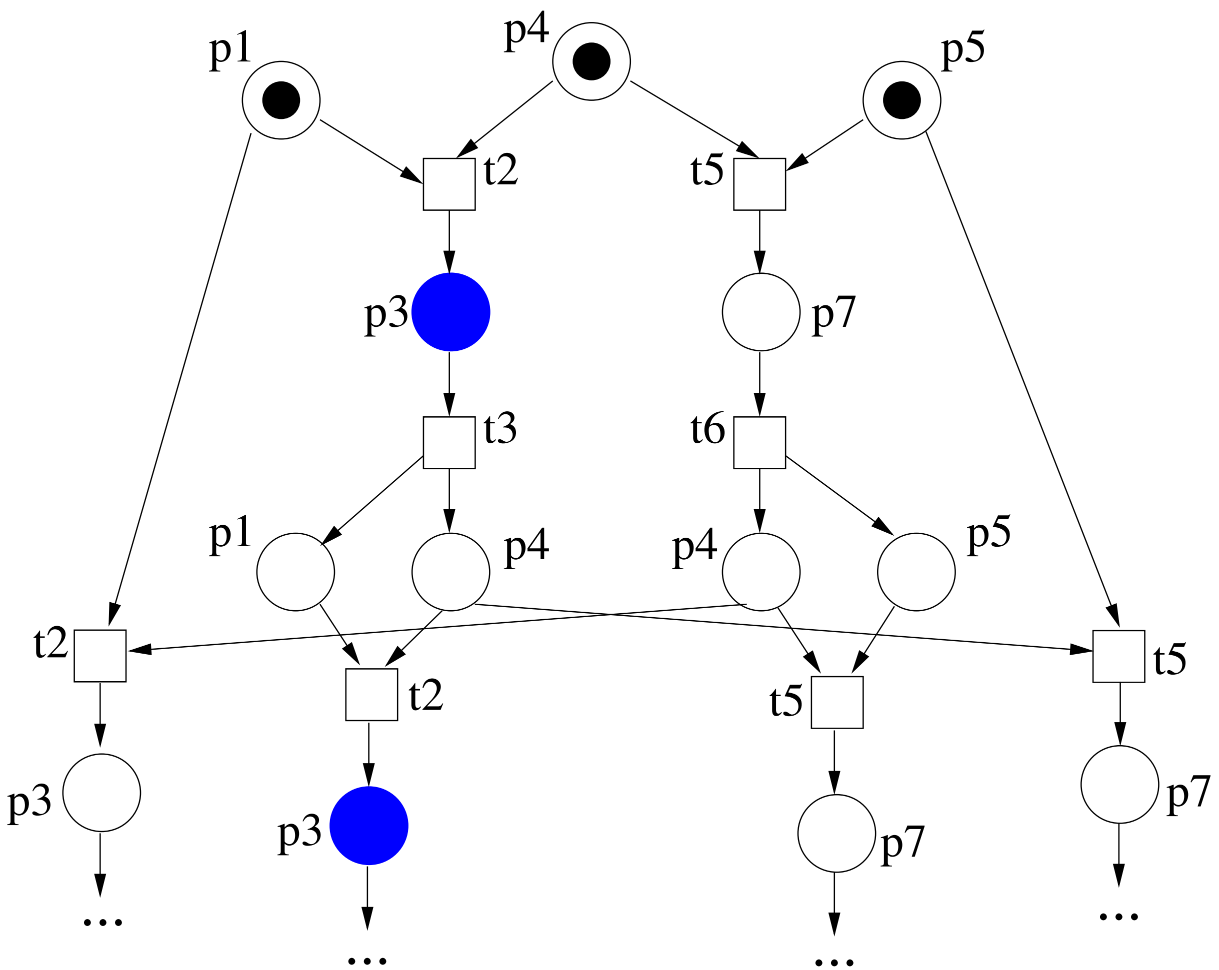
# Petri nets: Reviewing conflict, concurrency, . . .

In a non-contextual unfolding, any pair of places are either . . .



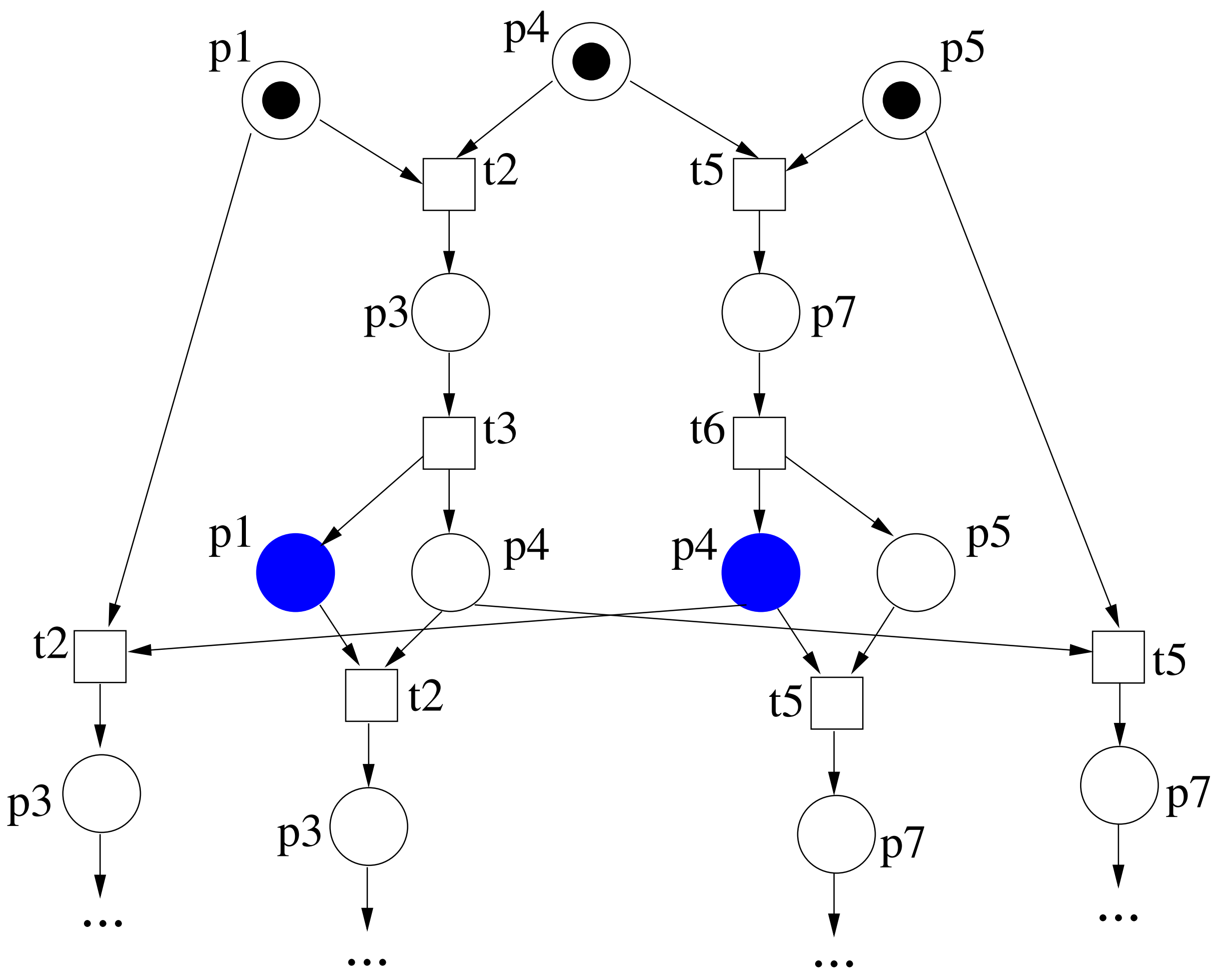
# Petri nets: Reviewing conflict, concurrency, etc

. . . in causal relationship (one must be consumed to produce the other), . . .



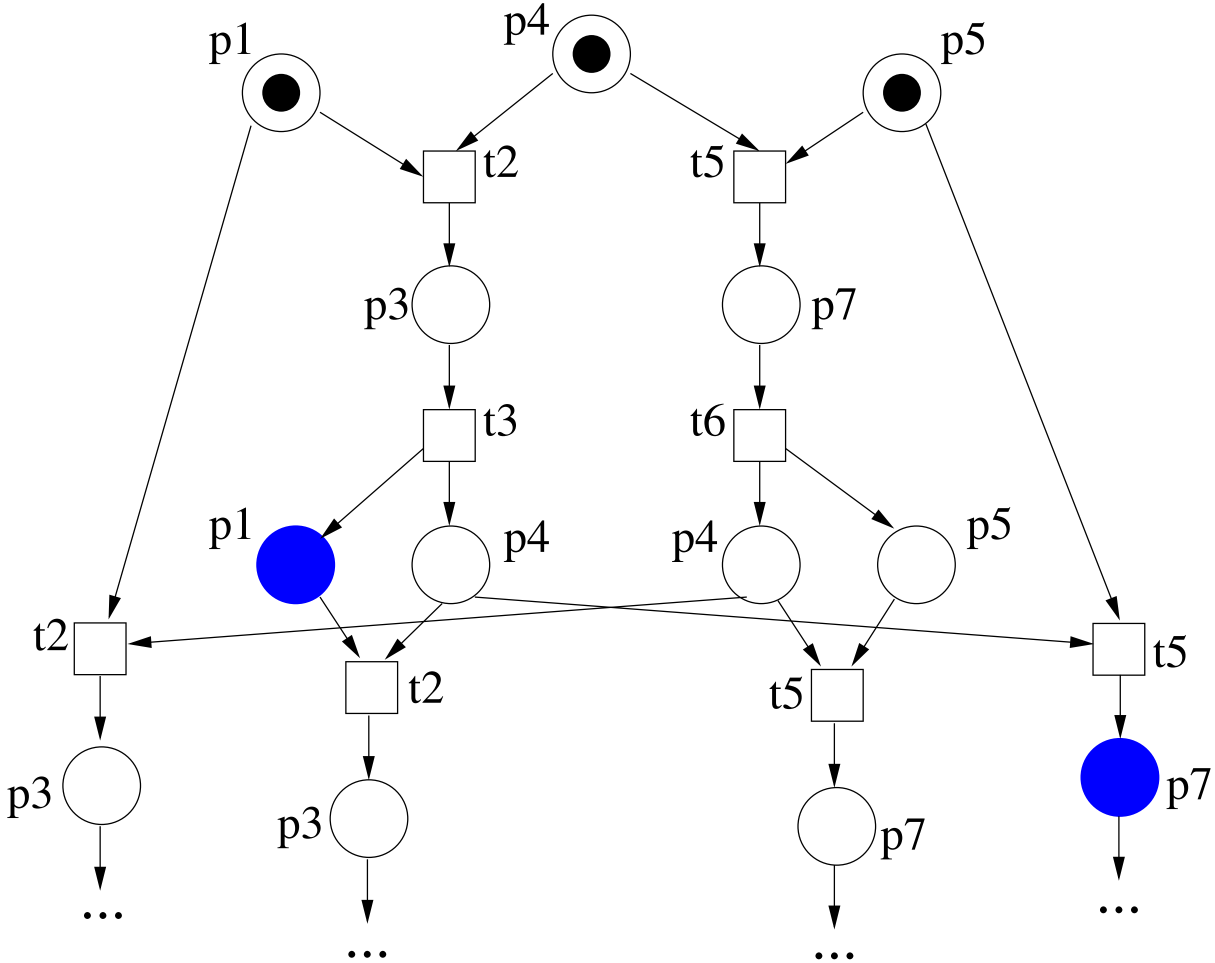
# Petri nets: Reviewing conflict, concurrency, etc

. . . or in conflict (must decide to generate one or the other), . . .



# Petri nets: Reviewing conflict, concurrency, etc

. . . or concurrent (can be marked at the same time).



# Petri nets: Causality and conflict

Let $x, y$ be two nodes (places or transitions) in a Petri net unfolding.

Let $<$ be the transitive closure of the relation $\{ (x, y) \mid x \in {}^\bullet y \}$.
$x$ is a cause of $y$ if $x < y$.

We write $\lfloor x \rfloor := \{ t \mid t$ is a transition s.t. $t \leq x \}$.

We say $x \# y$ ($x, y$ are in conflict) if there exist two distinct transitions $t, u$ such that $t \leq x$, $u \leq y$, and ${}^\bullet t \cap {}^\bullet u \neq \emptyset$.

We say $x \parallel y$ ($x, y$ are concurrent) if neither $x < y$, $y < x$, nor $x \# y$.

# Petri nets: Configurations and reachability

Let $C$ be a set of transitions in a Petri net unfolding.
We call $C$ be a configuration if

(i) $t \in C$ and $t' < t$ imply $t' \in C$ (i.e., $C$ is causally closed);

(ii) $t, t' \in C$ implies $\neg(t \# t')$ (i.e., $C$ is conflict-free)

A marking $M$ of the unfolding is reachable iff there exists a configuration $C$ s.t. $M = (M_0 \cup C^\bullet) \setminus {}^\bullet C =: M_C$, where $M_0$ is the initial marking.

Fact 1: A set $S$ of places in the unfolding is coverable iff

(i) $p \not< q$ for all pairs $p, q \in S$;

(ii) $D := \bigcup_{p \in S} \lfloor p \rfloor$ is a configuration (i.e., conflict-free).

Fact 2: Also, a set $S$ of places in the unfolding is coverable iff $p \parallel q$ for all $p, q \in S$.

# Petri nets: Deciding coverability

Using Fact 1:

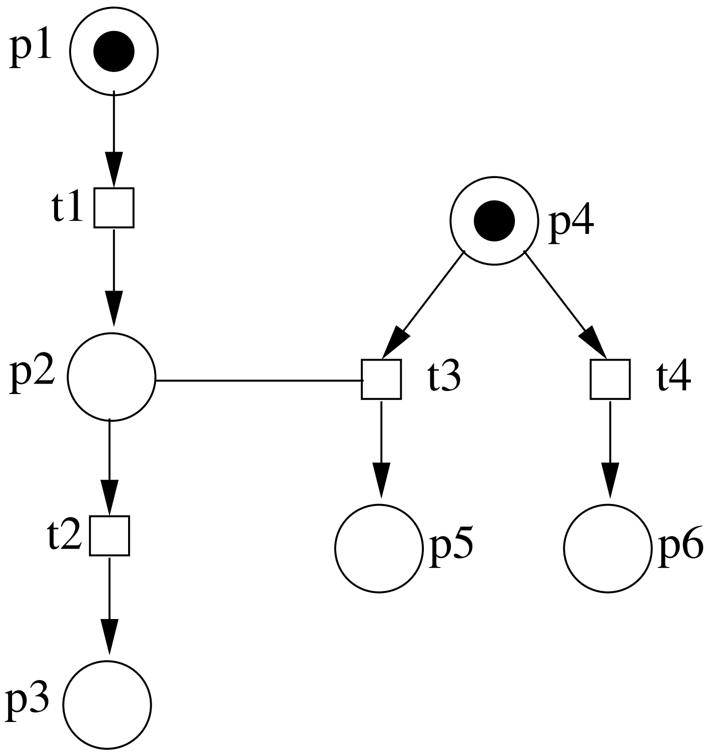
Linear marking algorithm on $D$; follow flow arcs backwards from $S$, check if places are consumed twice.

Using Fact 2:

Compute the pairwise $\|$ relation while constructing the unfolding (conflicts are "inherited" from causes).

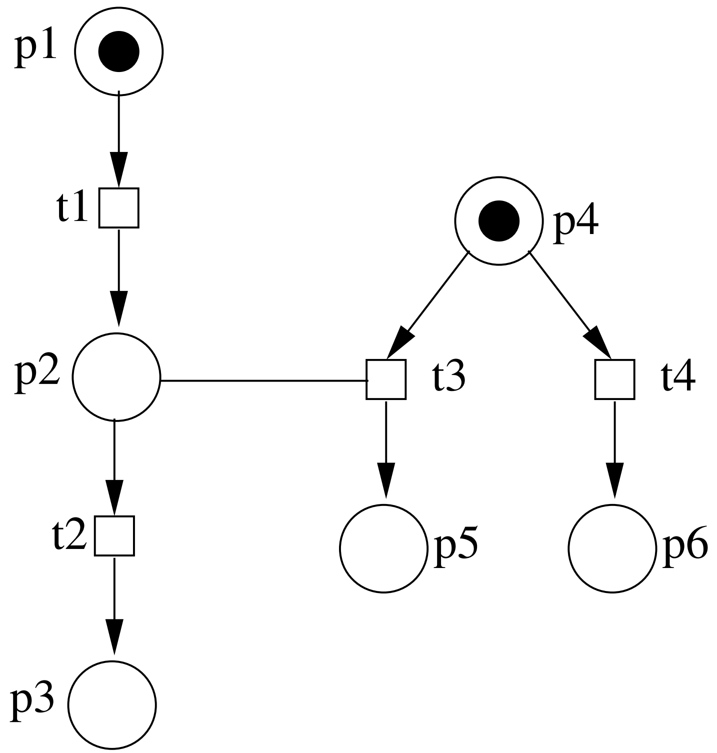
Time/space trade-off!

# Contextual nets: How to adapt these notions?




Consider the contextual net shown above. How do we adequately treat the read arcs in the causality and conflict relations?

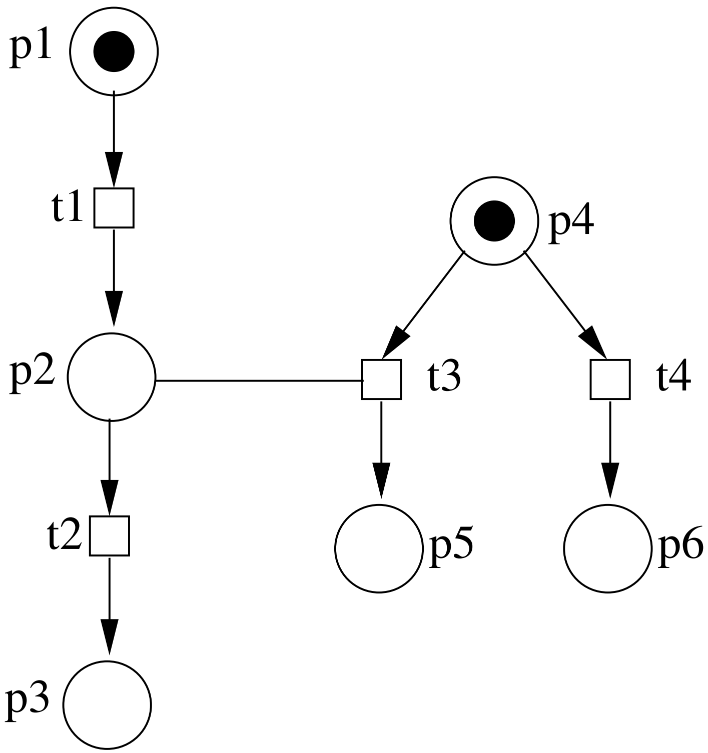
# Contextual nets: Adapting the notion of causality




Let us re-formulate $x < y$ as "$x$ must necessarily occur before $y$."

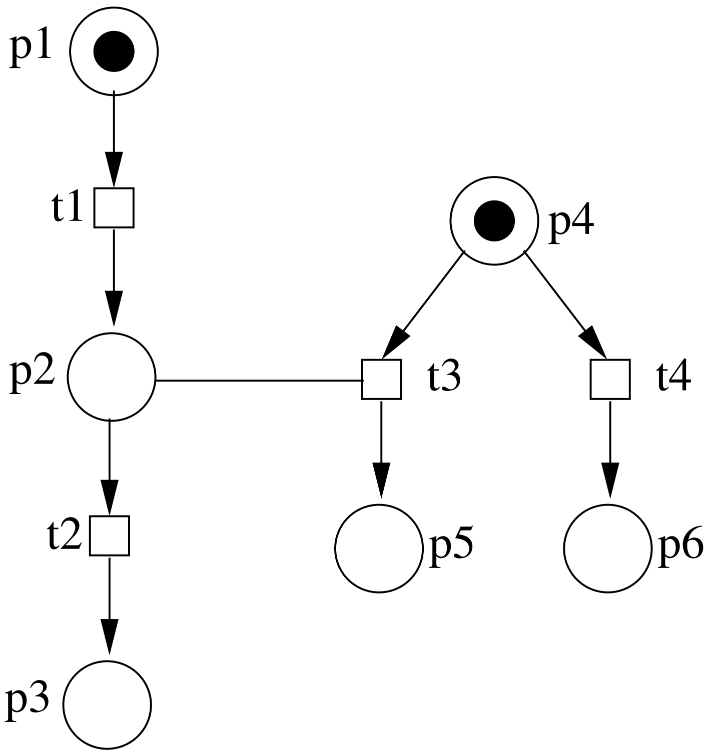
Then we have (as usual) $p_1 < t_1$ and $t_1 < p_2$, but also $t_1 < t_3$.

# Contextual nets: Adapting the notion of conflict




$t_2$ and $t_3$ are in a special relation; no conflict in the usual sense, since both can happen, but $t_3$ must happen first.

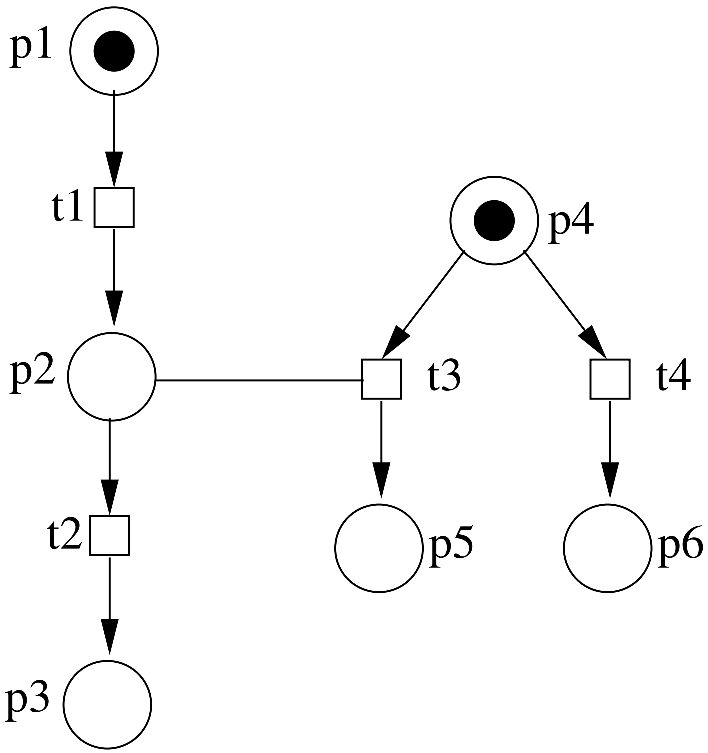
# Contextual nets: Adapting the notion of conflict




We introduce the notion of asymmetric conflict from *x* to *y*:

"If both *x* and *y* happen, then *x* happens first."

# Contextual nets: Adapting the notion of conflict




This notion generalizes causality and (symmetric) conflict, i.e. we have $t_1 \nearrow t_2$, $t_1 \nearrow t_3$ (causes), $t_3 \nearrow t_2$ (asymm. conflict), $t_3 \nearrow t_4$, and $t_4 \nearrow t_3$ (normal conflict).

# Contextual nets: Causality and conflict

Let $<$ be the least transitive relation satisfying

$s < t$ if $s$ is a place, $t$ a transition, and $s \in {}^\bullet t$;

$t < s$ if $s$ is a place, $t$ a transition, and $s \in t^\bullet$;

$t < t'$ if $t$ and $t'$ are transitions, and $t^\bullet \cap \underline{t'} \neq \emptyset$.

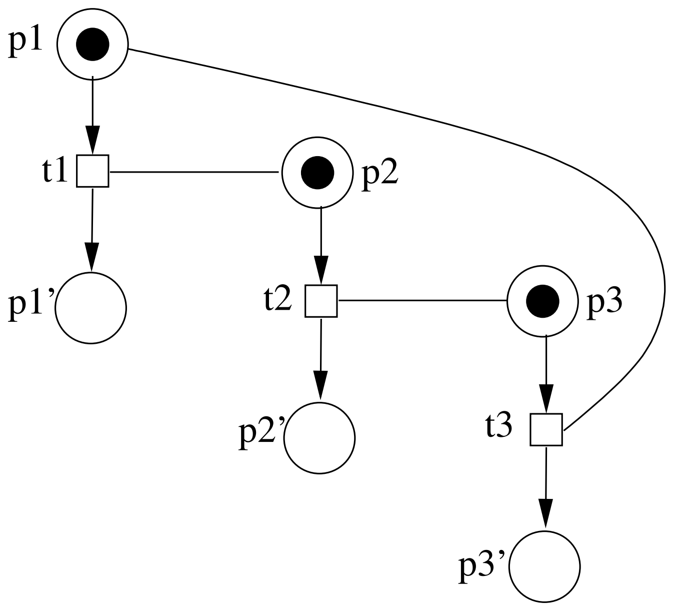
$\lfloor x \rfloor := \{ t \mid t$ is a transition s.t. $t \leq x \}$ as before.

Let $t, t'$ be distinct transitions. They are in asymmetric conflict, written $t \nearrow t'$ iff $t < t'$, or ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$, or $\underline{t} \cap {}^\bullet t' \neq \emptyset$.

# Asymmetric conflicts

An asymmetric conflict $t \nearrow t'$ can be seen as a scheduling constraint; a cycle in the $\nearrow$ relation indicates that the transitions involved cannot all occur together.

If no read arcs are present, then all simple cycles are of length 2.
However, read arcs can lead to longer cycles, as the example below shows:
The net is identical to its unfolding, and we have $t_1 \nearrow t_2 \nearrow t_3 \nearrow t_1$.



# Contextual nets: Configurations and reachability

Let $C$ be a set of transitions in a contextual unfolding.
We call $C$ a configuration iff:

(i) $t \in C$ and $t' < t$ imply $t' \in C$ (i.e., $C$ is causally closed);

(ii) $\nearrow \cap (C \times C) =: \nearrow_C$ does not contain any cycles;

(iii) $\{ t' \in C \mid t' \nearrow t \}$ is finite for all $t \in C$.

As before, a marking $M$ of the unfolding is reachable iff there exists a configuration $C$ s.t. $M = (M_0 \cup C^\bullet) \setminus {}^\bullet C$, where $M_0$ is the initial marking.

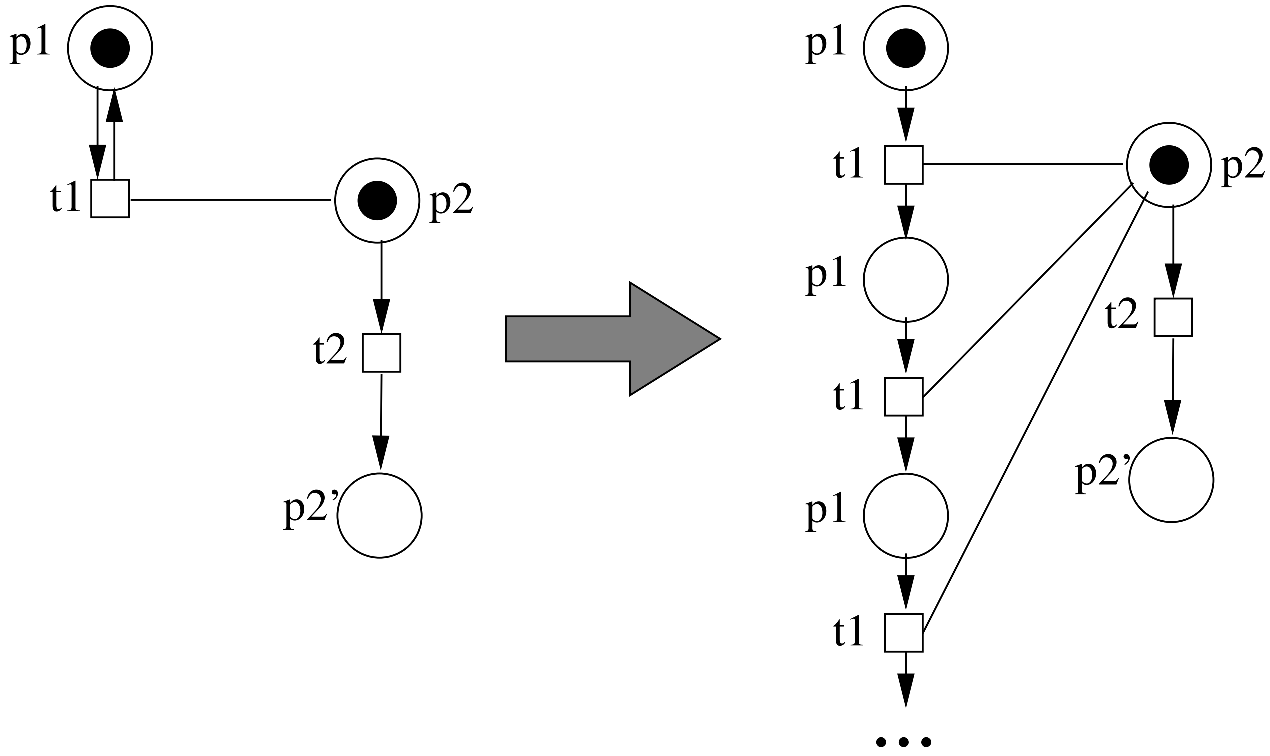
Fact 1′: A set of places $S$ in the unfolding is coverable iff

(i) $p \not< q$ for all $p, q \in S$;

(ii) $D := \bigcup_{p \in S} \lfloor p \rfloor$ is a configuration (i.e., absence of conflict cycles).

# Why condition (iii) for configurations is necessary

Consider the net below (left) and its unfolding (right):




The set $C'$ of all transitions in the unfolding fulfils conditions (i) and (ii) but not (iii). Indeed, there is no firing sequence containing all transitions in $C'$ – when would one fire $t_2$?

# Petri nets: Deciding coverability

Using Fact 1′:

Linear algorithm on $D$; perform DFS on the graph given by $\nearrow_D$, search for a cycle. (Good news!)

# Petri nets: Deciding coverability

Using Fact 1′:

Linear algorithm on $D$; perform DFS on the graph given by $\nearrow_D$, search for a cycle. (Good news!)

Using Fact 2′:

Wait. . . this doesn't exist. (Bad news!)

# Petri nets: Deciding coverability

Using Fact 1′:

Linear algorithm on $D$; perform DFS on the graph given by $\nearrow_D$, search for a cycle. (Good news!)

Using Fact 2′:

Wait. . . this doesn't exist. (Bad news!)

Indeed, a binary relation is not sufficient to detect cycles.

Mitigating factors:

Symmetric conflicts can still be handled in the same way.

Absence of non-symmetric cycles in the net implies their absence in the unfolding.

Other tricks. . .?

# Complete finite prefixes

In general, unfoldings are infinite objects. We are interested in computing just a finite part of them that contains all “relevant” information (in this case, all reachable markings).

For (normal) Petri nets, this is achieved by introducing cut-offs.

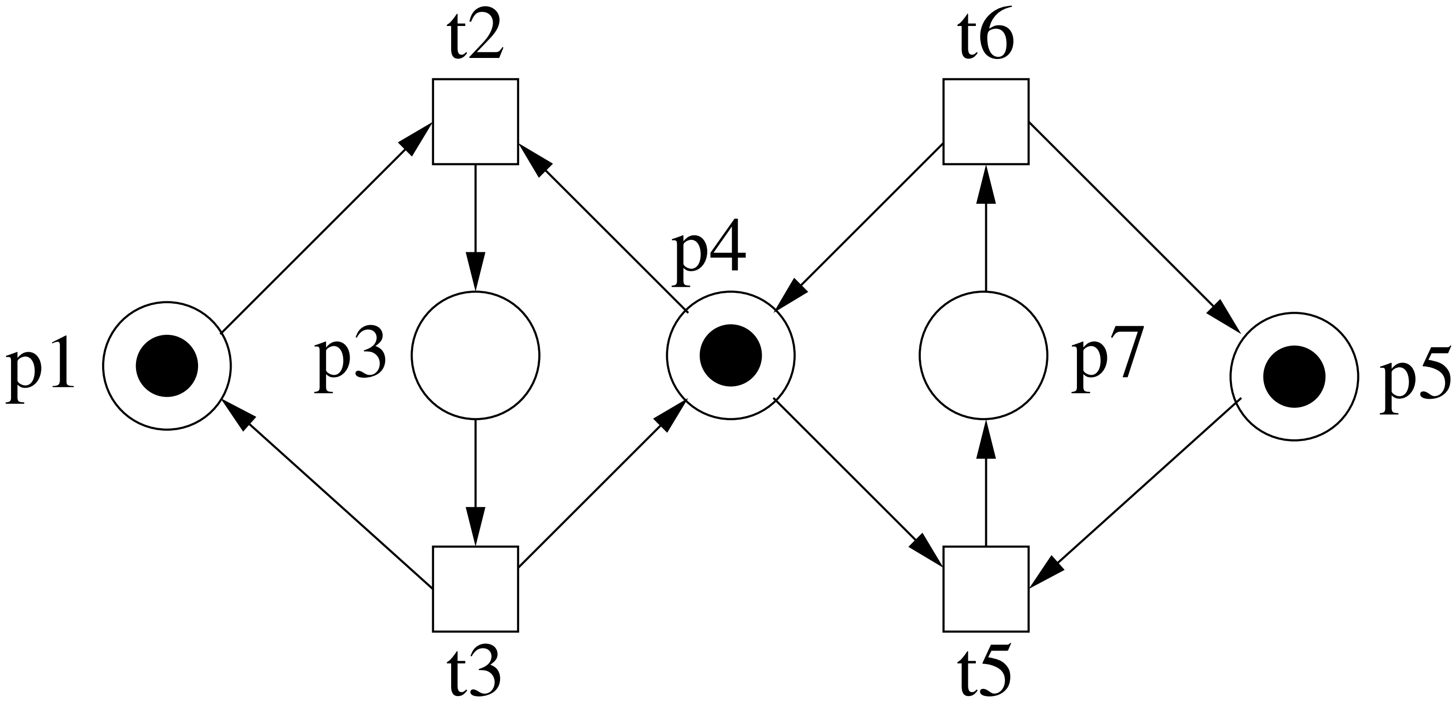
One introduces a partial order on unfolding transitions $\prec$ that refines $<$.

The unfolding prefix is generated by adding one transition at a time, respecting $\prec$; with every transition $t$, we associate the marking $M_{\lfloor t \rfloor}$.

If $M_{\lfloor t \rfloor}$ equals the initial marking, or if there is a transition $t' \prec t$ with $M_{\lfloor t \rfloor} = M_{\lfloor t' \rfloor}$, then $t$ is declared a cut-off.

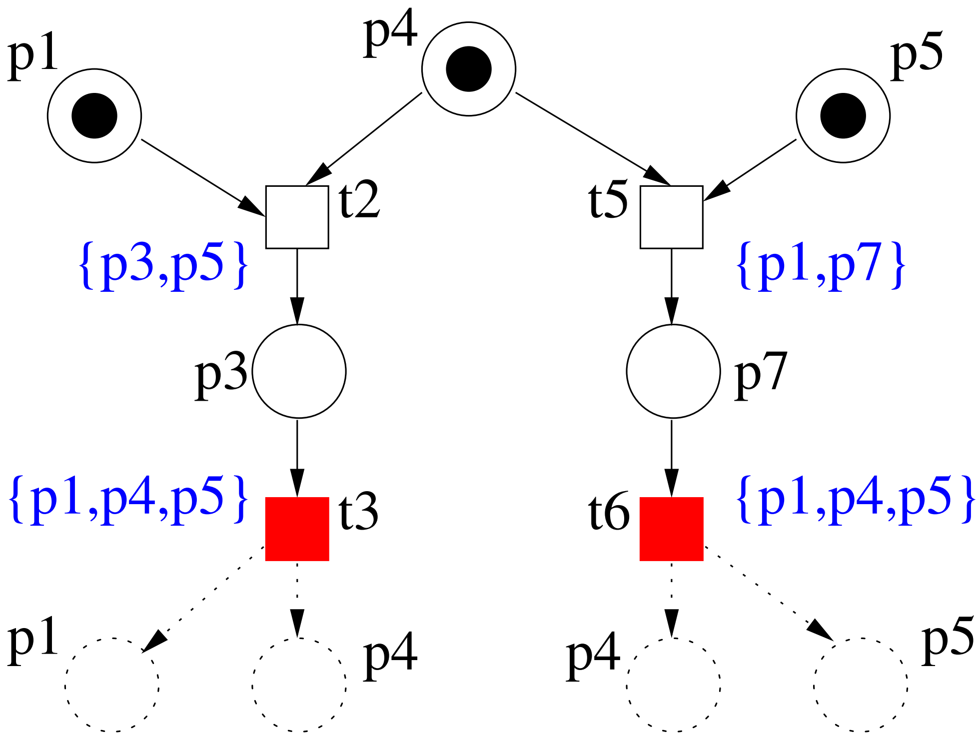
Output places of cut-offs are not considered for further additions.

# Example: Petri net (again). . .



# . . . and a prefix of its unfolding

Below, each transition is annotated with its marking.




Cut-offs are marked in red; the prefix is complete.

# Adequate orders

The unfolding procedure with cut-offs does not yield a complete prefix for every order $\prec$ (see, e.g., [EKS08]).

However, it a complete prefix *is* produced provided that $\prec$ satisfies certain conditions:

McMillan's condition: $|\lfloor t \rfloor| < |\lfloor t' \rfloor|$ implies $t \prec t'$

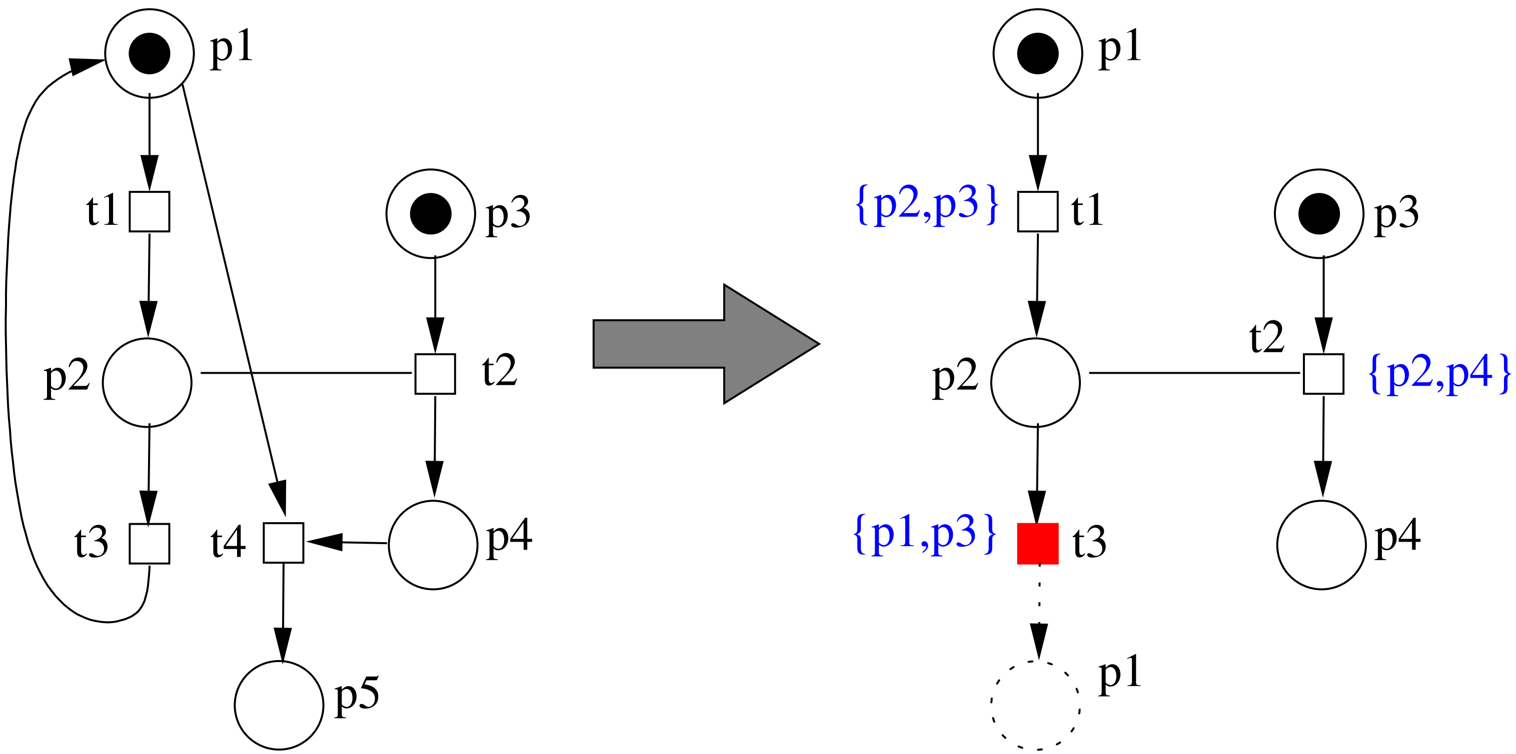
Esparza/Römer/Vogler conditions: $\prec$ must be well-founded, $\lfloor t \rfloor \subset \lfloor t' \rfloor$ implies $t \prec t'$, $\prec$ is "preserved by finite extensions".

Note: The ERV conditions give rise to smaller prefixes than McMillan's.

Another note: Cut-offs have no effect on the exponential blowups described earlier!

# Cut-offs for contextual prefixes

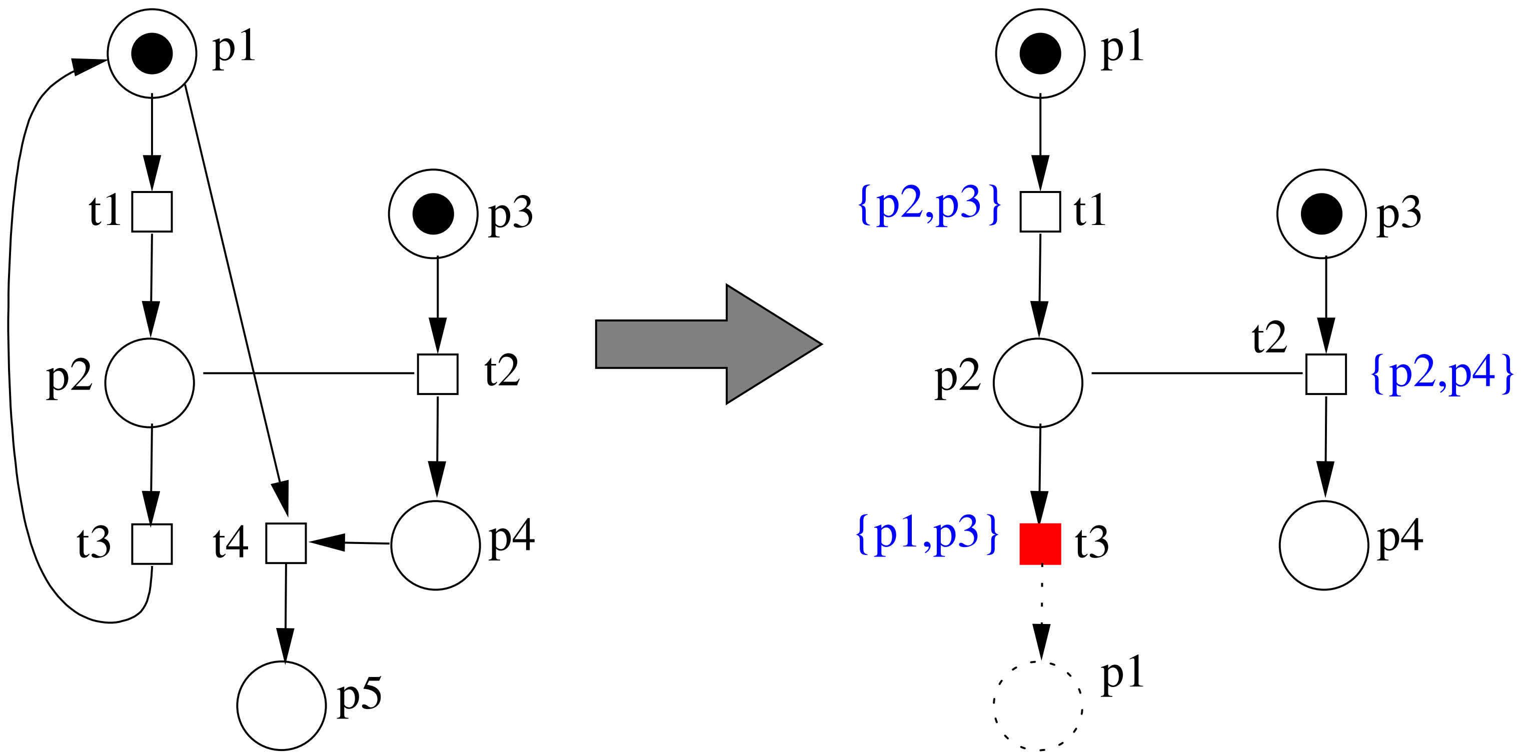
For contextual nets, the cut-off procedure cannot be directly applied:




The occurrence of $t_3$ in the unfolding is declared a cut-off;
therefore, $t_4$ is never added (even though $t_1 t_2 t_3 t_4$ can be fired in the net).

# Cut-offs for contextual unfoldings

For contextual nets, the cut-off procedure cannot be directly applied:




Intuitively, $t_3$ occurs in two different situations: with and without the occurrence of $t_2$. The conventional cut-off method ignores the contribution of the latter.

# Histories

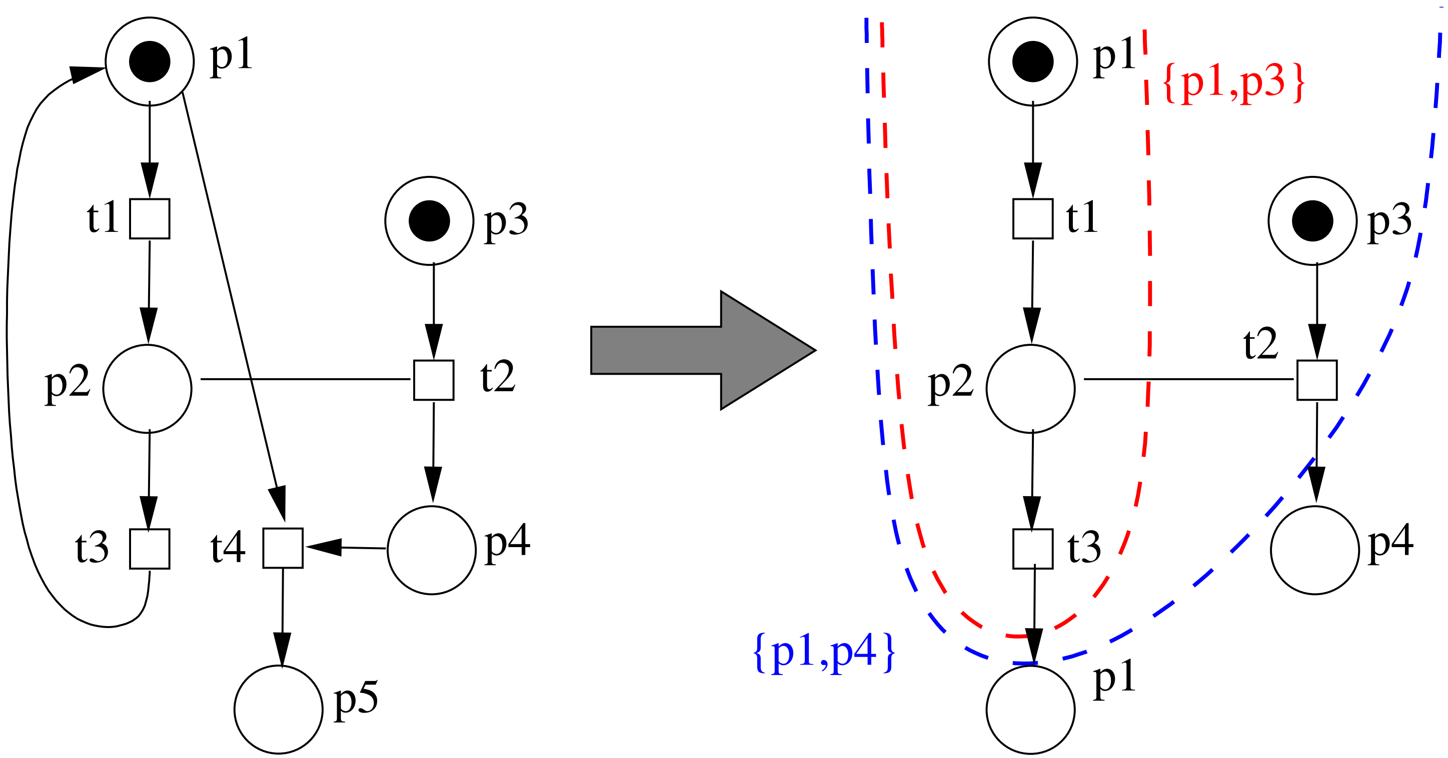
We formalize this intuition in the notion of histories:

Let $C$ be a configuration and $t \in C$ a transition. The history of $t$ in $C$ is the configuration $C[\![t]\!] := \{ t' \in C \mid t' \nearrow_C t \}$.

We call $Hist(t)$ the set of all histories that $t$ has in the unfolding. (Generally, these histories differ in the set of “reading” transitions they contain.)

# Example: Histories

Below, two histories for $t_3$ and their markings are shown:



# Histories

We formalize this intuition in the notion of histories:

Let $C$ be a configuration and $t \in C$ a transition. The history of $t$ in $C$ is the configuration $C[\![t]\!] := \{\, t' \in C \mid t' \nearrow_C t \,\}$.

We call $Hist(t)$ the set of all histories that $t$ has in the unfolding. (Generally, these histories differ in the set of “reading” transitions they contain.)

The conventional cut-off scheme considers only one history, $\lfloor t \rfloor$, which is too restrictive. On the other hand, considering all histories is fraught with problems (infinitely many, not constructive). (Approaches by Vogler et al and Winkowski.)

Solution: Identify a finite subset of “relevant” histories for cut-off selection.

# Enriched prefix

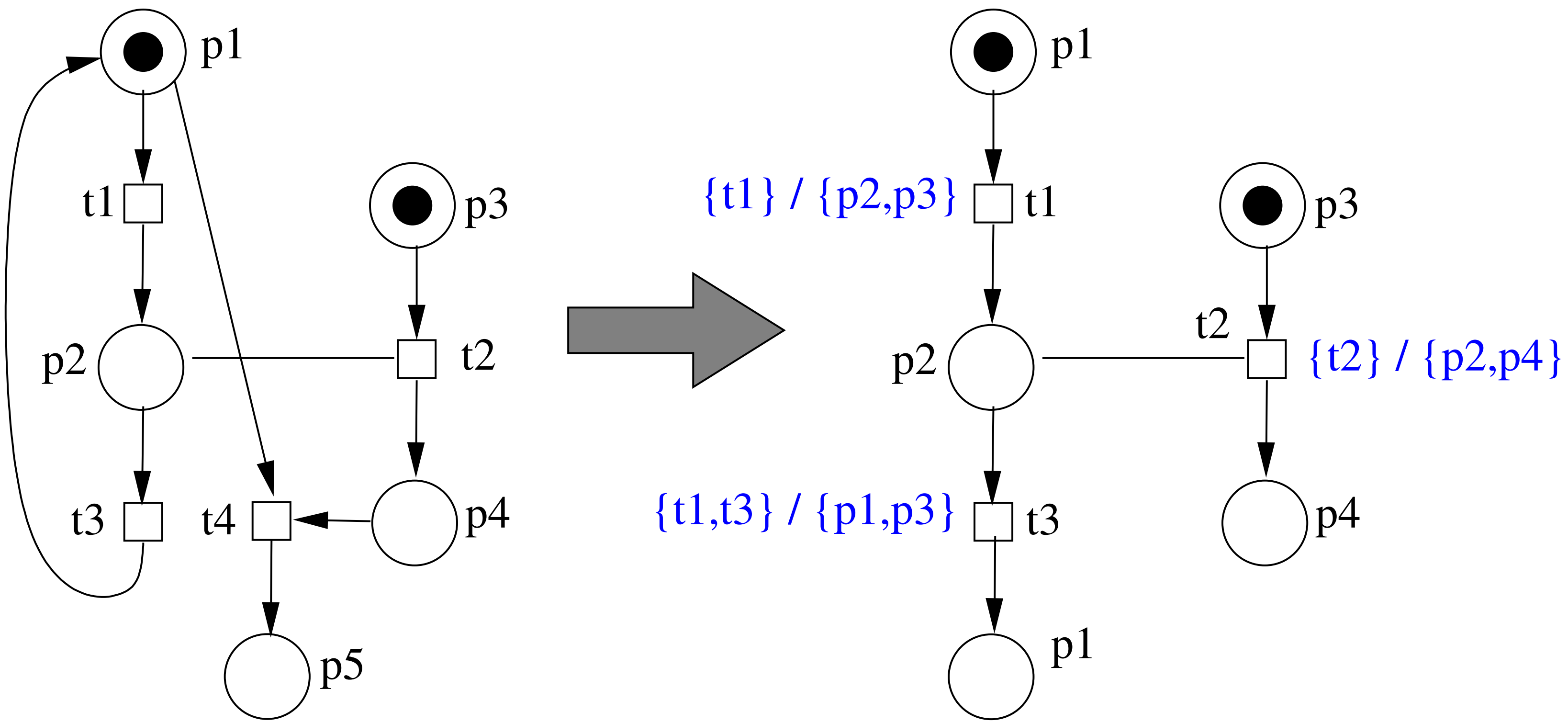
We shall lift the notion of cut-offs from transitions to histories.

To this end we introduce the notion of an enriched prefix, which is a prefix of the unfolding in which every transition is labelled with a subset of its histories. We assign the marking $M_C$ to each pair $\langle t, C \rangle$.

.

# Example: Enriched prefix

An example of an enriched prefix is shown below (histories/markings in blue):



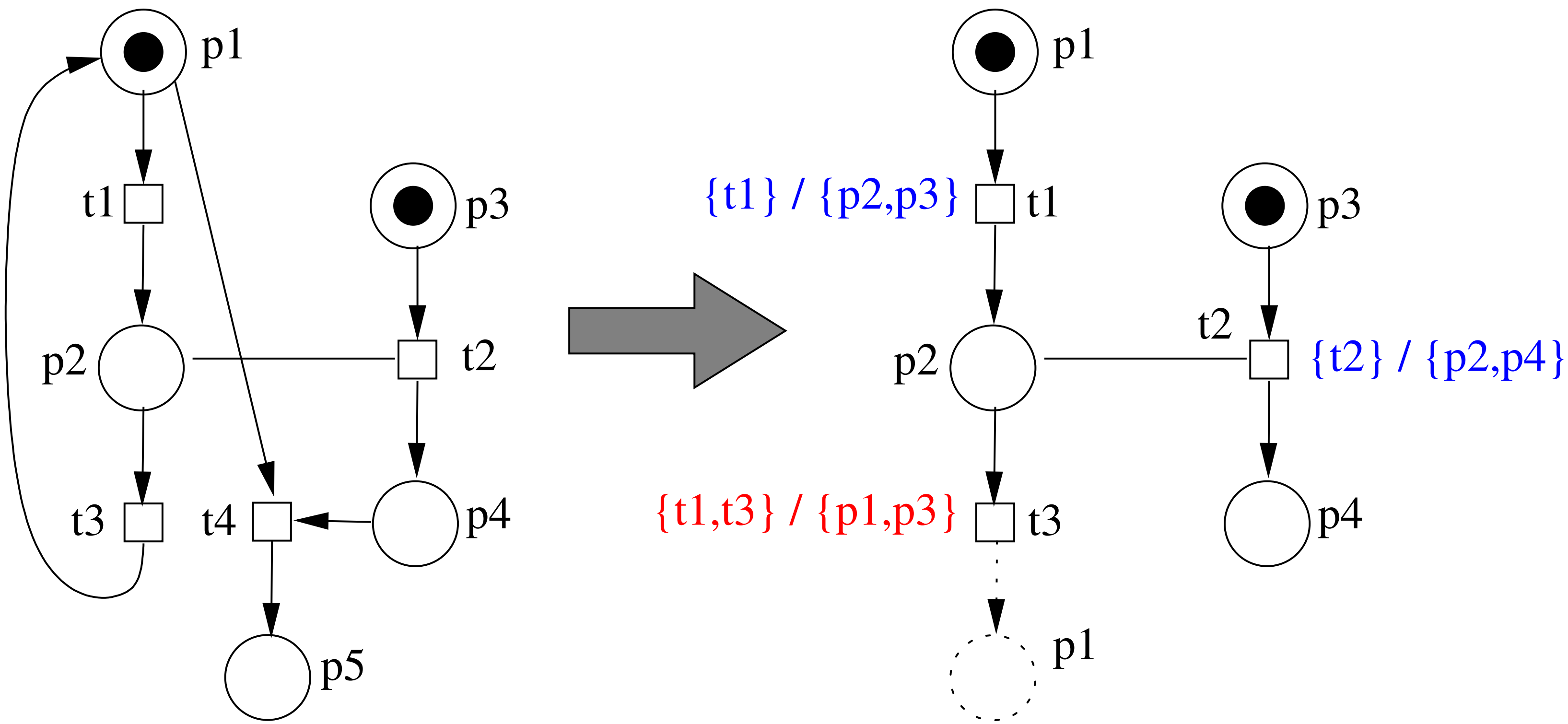
# Enriched prefix

We shall lift the notion of cut-offs from transitions to histories.
To this end we introduce the notion of an enriched prefix, which is a prefix of the unfolding in which every transition is labelled with a subset of its histories. We assign the marking $M_C$ to each pair $\langle t, C \rangle$.

Likewise, the ordering $\prec$ is lifted to histories. Now, a pair $\langle t, C \rangle$ is called a cut-off (in the *enriched* prefix) if its marking $M_C$ is identical to the initial marking, or there exists a pair $\langle t', C' \rangle \prec \langle t, C \rangle$ with $M_{C'} = M_C$.

# Example: Enriched prefix

An example of an enriched prefix is shown below (histories/markings in blue):




Here, the pair $\langle t_3, \{t_1, t_3\}\rangle$ is a cut-off.

# Enriched prefix

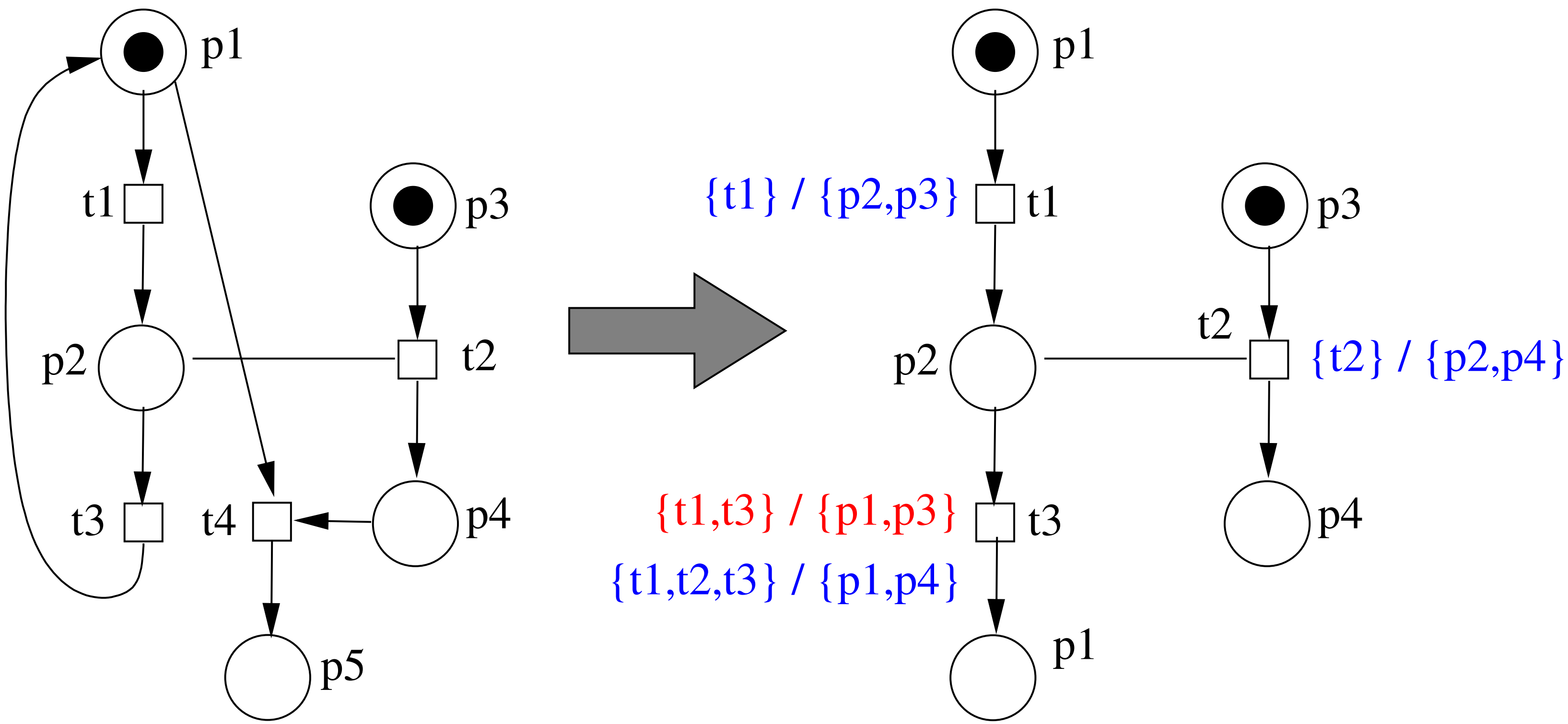
We shall lift the notion of cut-offs from transitions to histories.
To this end we introduce the notion of an enriched prefix, which is a prefix of the unfolding in which every transition is labelled with a subset of its histories. We assign the marking $M_C$ to each pair $\langle t, C \rangle$.

Likewise, the ordering $\prec$ is lifted to histories. Now, a pair $\langle t, C \rangle$ is called a cut-off (in the *enriched* prefix) if its marking $M_C$ is identical to the initial marking, or there exists a pair $\langle t', C' \rangle \prec \langle t, C \rangle$ with $M_{C'} = M_C$.

Output places of transition $t$ will be considered for additions only if $t$ is labelled with at least one non-cut-off history.

# Example: Enriched prefix

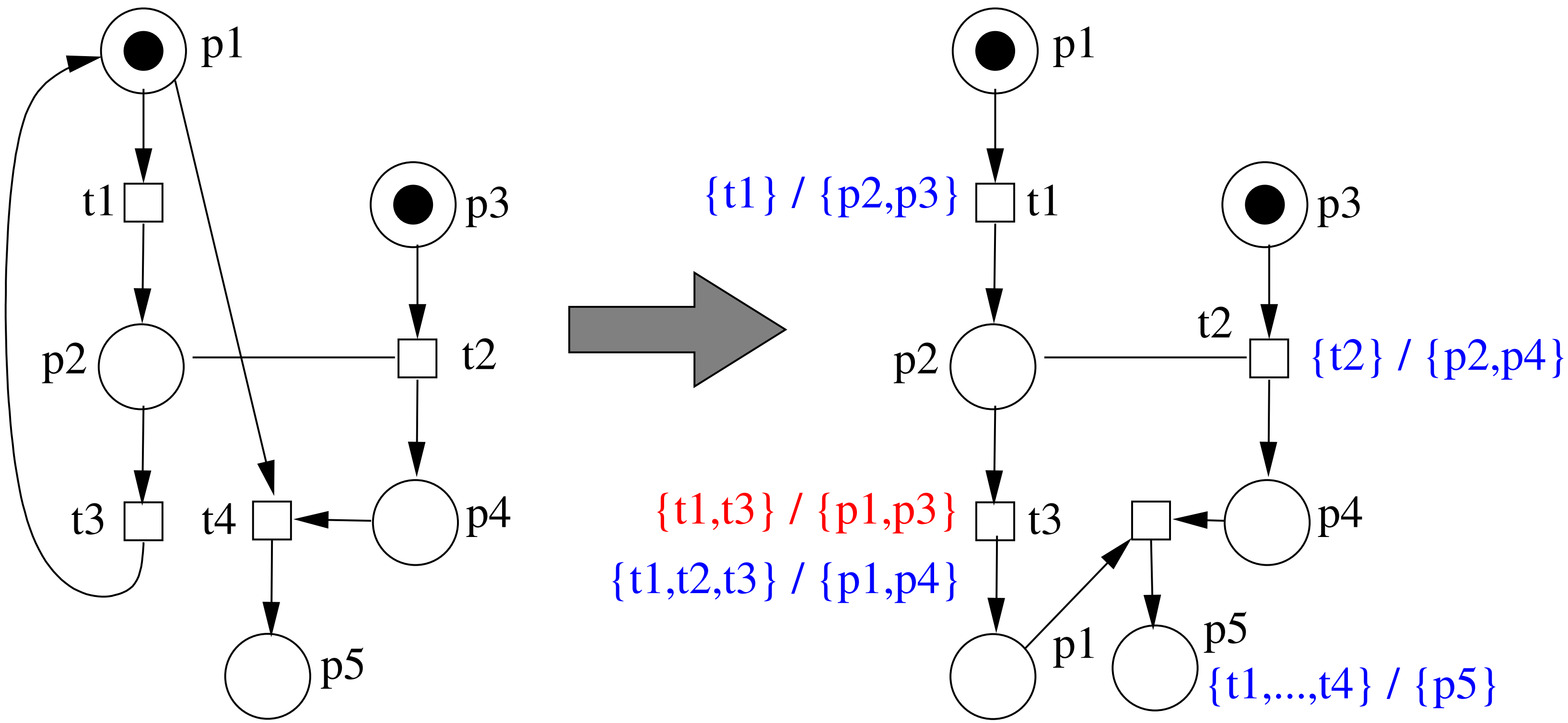
An example of an enriched prefix is shown below (histories/markings in blue):




Notice that the pair $\langle t_3, \{t_1, t_2, t_3\}\rangle$ is *not* a cut-off!

# Example: Enriched prefix

An example of an enriched prefix is shown below (histories/markings in blue):




Hence, the second copy of $p_1$ can be considered for extensions, allowing for $t_4$.

# Closed enriched prefixes

Problem: How do we make this effective, i.e. how does one construct and choose the histories, and which orderings lead to a complete prefix?

An enriched prefix is closed if for each pair $\langle t, C\rangle$ s.t. $C$ is contained in the labelling of $t$, the following holds:

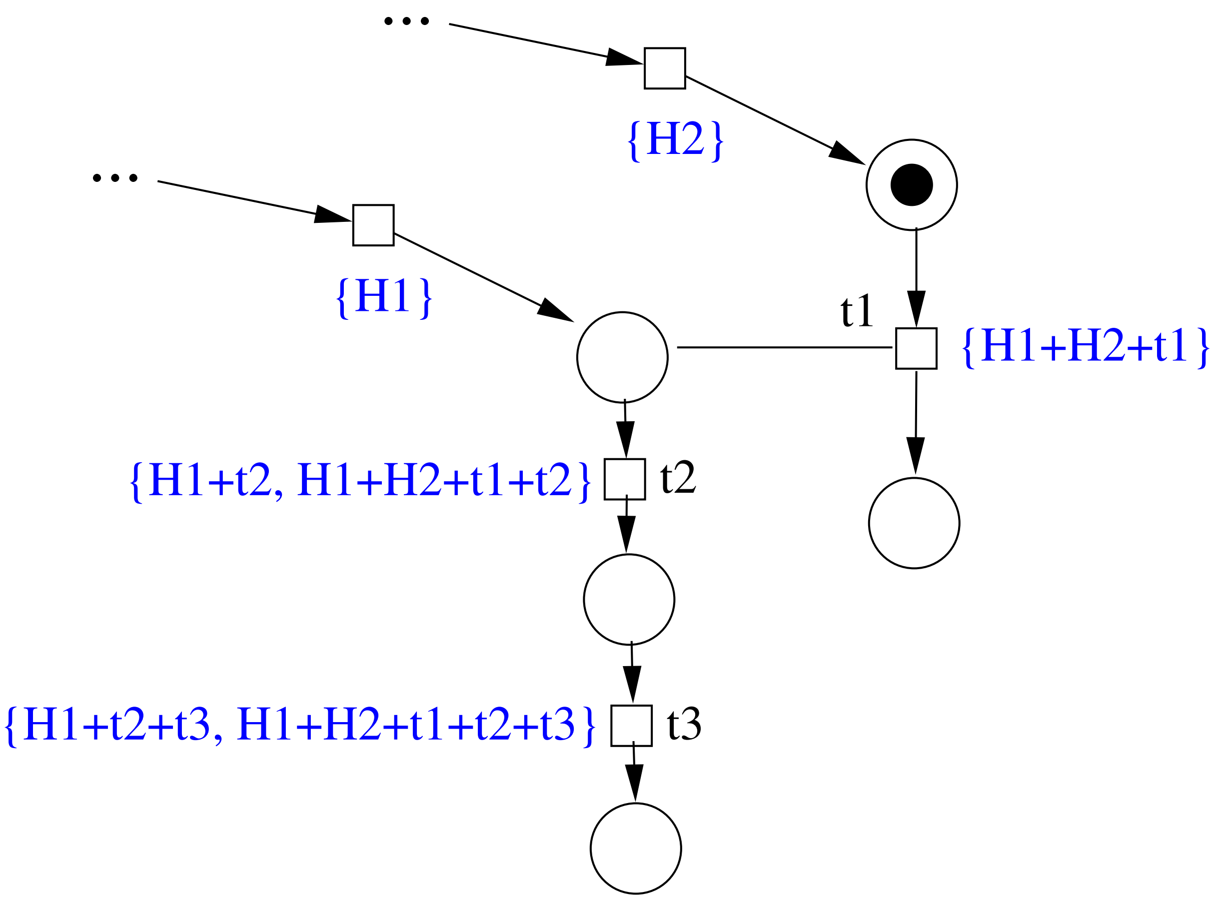
> If $t' \in C$, then $C[\![t']\!]$ is in the labelling of $t'$.

For $\prec$, we consider adequate orders (lifted to histories).

Considering only closed prefixes kills two flies/moskitoes with one strike:

> Candidates for additional histories of some transition $t$ are always constructed by *uniting* histories labelling *direct* $\nearrow$-predecessors of $t$.
>
> One can store those histories in memory by simply pointing to those histories.

# Example: Closed prefix



# Prefix ordering

Let $E_1, E_2$ be two enriched prefixes. We write $E_1 \sqsubseteq E_2$ if the net structure of $E_1$ is a prefix of that of $E_2$, and all tuples $\langle t, C \rangle$ from $E_1$ are also in $E_2$.

Note: When constructing a prefix, adding a transition or a pair $\langle t, C \rangle$ moves us "upwards" in $\sqsubseteq$.

Lemma: The set of closed prefixes with $\sqsubseteq$ forms a complete lattice.

# Constructing a complete prefix

Theorem: Given an adequate order $\prec$, there is a maximal closed prefix without cut-offs that is complete.

This gives us a strategy for constructing a complete prefix:

Start with the minimal prefix (i.e., copies of the initial marking).

Among all transitions and their additional histories that can be constructed from direct $\nearrow$-predecessors, pick a $\prec$-minimal one that is not a cut-off, and add it.

Continue until no such additions are possible.

# Efficiency issues / Comparison with PR approach

Final contextual unfolding up to exponentially smaller than conventional Petri-net unfolding using the PR approach. (Good.)

Hidden complexity: For 1-safe nets, we get one $\langle t, C\rangle$ tuple for every transition in the PR unfolding. (Not so nice.) For general $n$-bounded nets, the memory requirements are smaller for the contextual unfolding. (Good.)

Even for 1-safe nets, deciding coverability from the contextual unfolding is easier than from the PR unfolding. (Good!)

Even for 1-safe nets, the contextual prefix construction *may* be quicker because we have additional knowledge about the relationship between histories. (hopefully good)

Future work: implementation in MOLE.

Questions?