Slicing Petri Nets with an Application to Workflow Verification

Astrid Rakow
astrid.rakow@informatik.uni-oldenburg.de

TrustSoft Graduate School, University of Oldenburg, Germany

SOFSEM 2008

1 Outline

Petri Nets

Slicing Algorithm

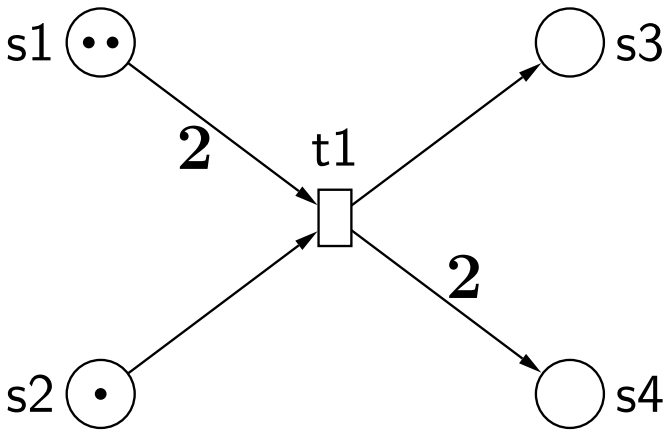
Results

Reducible Nets

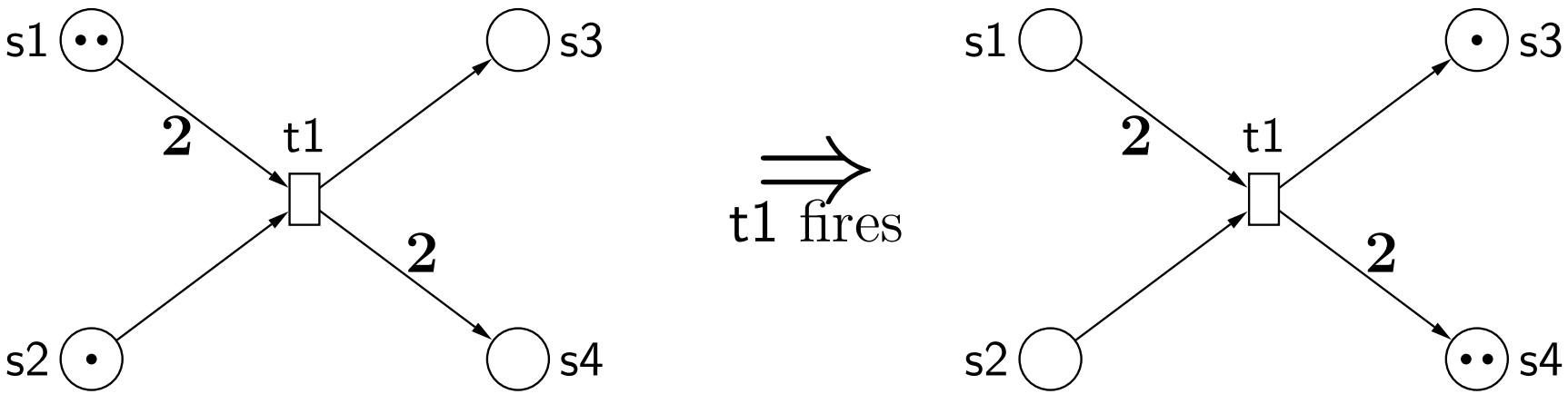
- Workflow Nets

1 Petri Net

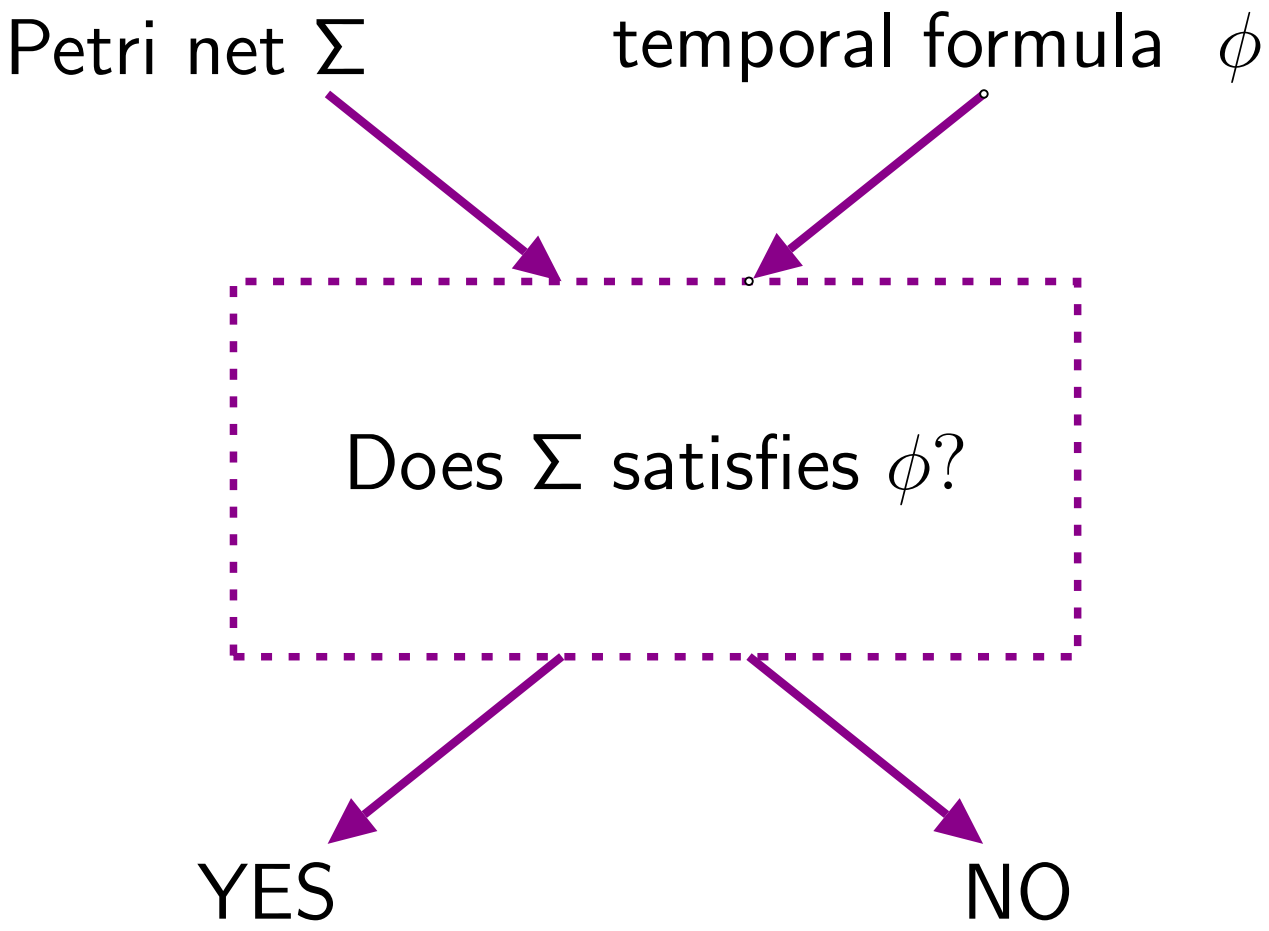
$$\Sigma = (S, T, W, M_0)$$



1 Petri Net



1 Model Checking



2 Slicing

Aim: Tackle the State Explosion Problem for Model Checking

3 Slicing

Aim: Tackle the State Explosion Problem for Model Checking

Given a marked **net Σ**, find a **subnet Σ'** such that

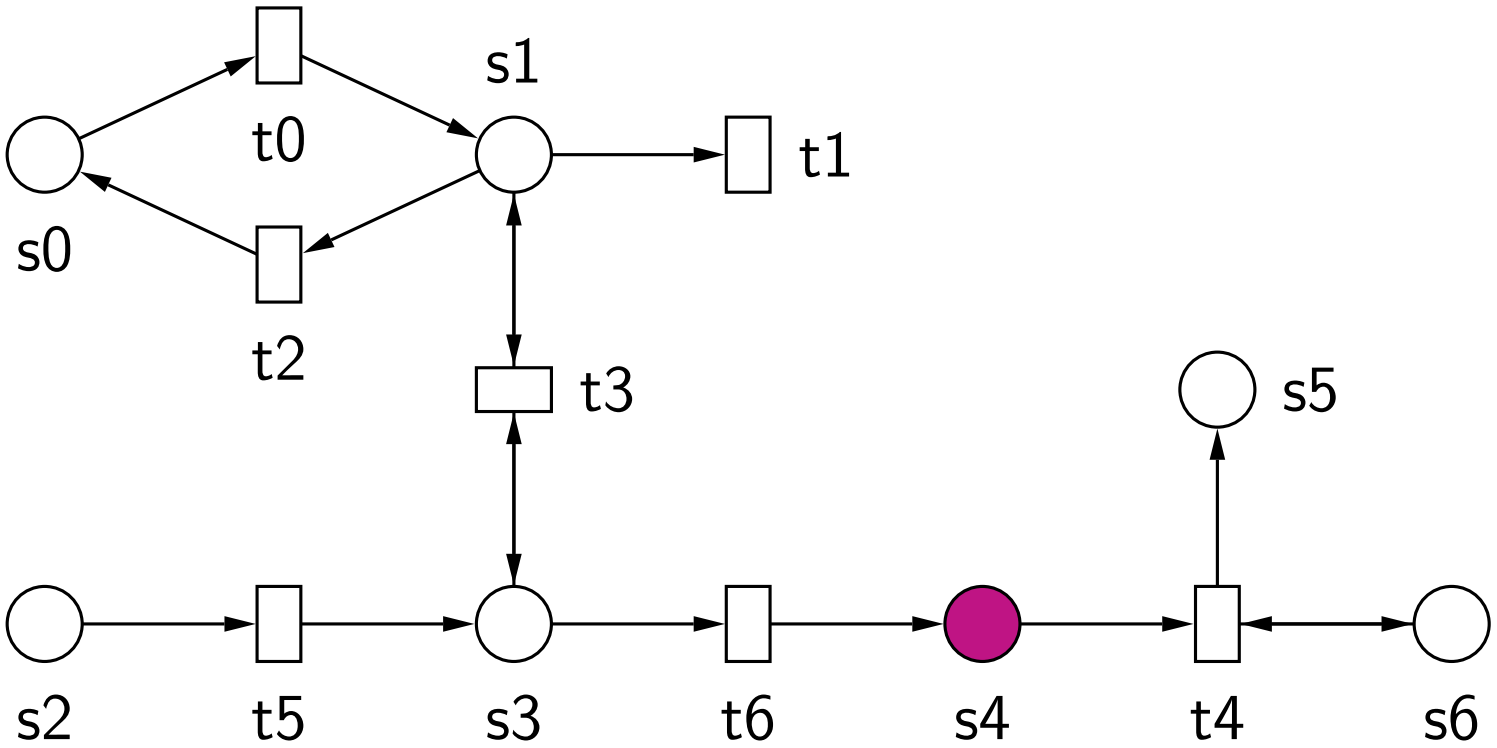
(i) $\Sigma \models \phi \Rightarrow \Sigma' \models \phi$ and $\Sigma' \models \phi \Rightarrow \Sigma \models \phi$

(falsification) (verification)

for a CTL* formula ϕ and

(ii) Σ' has a **smaller state space** than Σ.

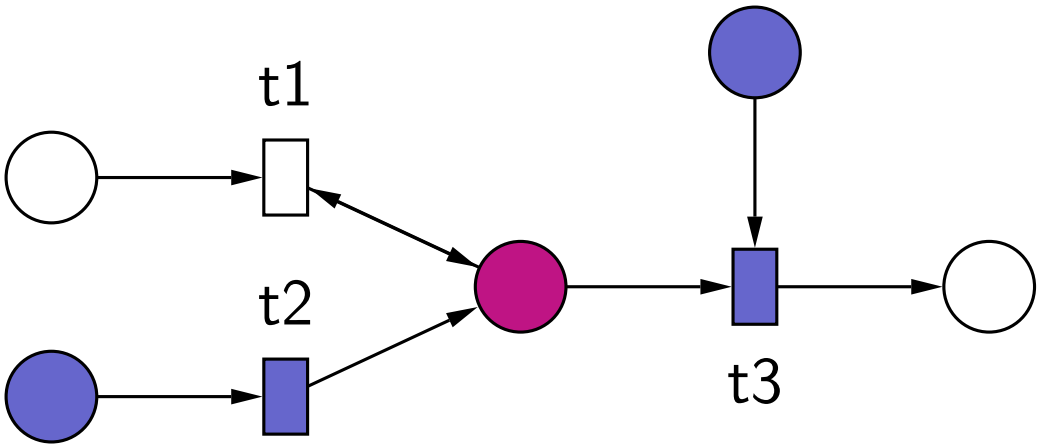
4 What do we need to keep



.. if we are interested in whether $\mathbf{A\,FG}\,(s4, 1)$ holds?

4 What do we need to keep?

Basic Idea:

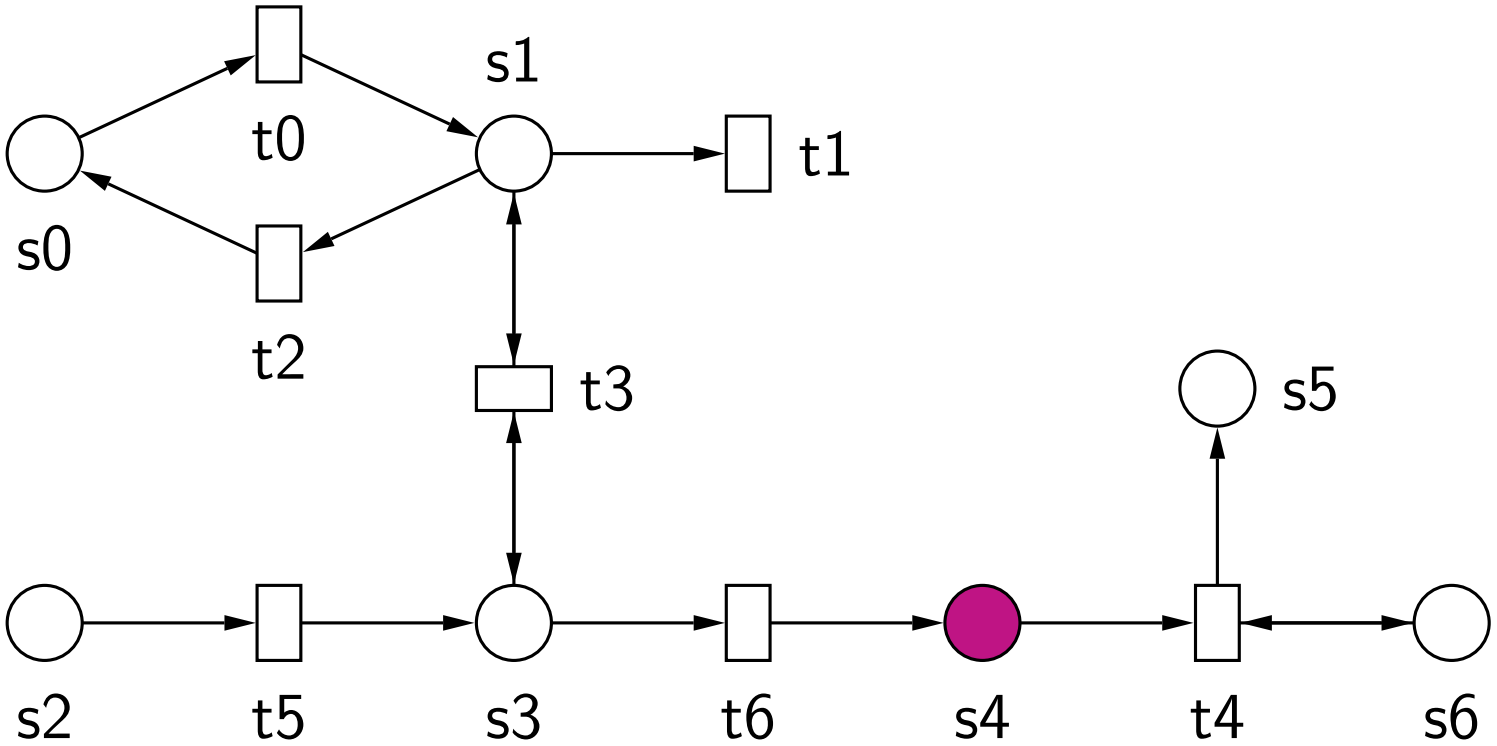


5 Slicing a Petri Net

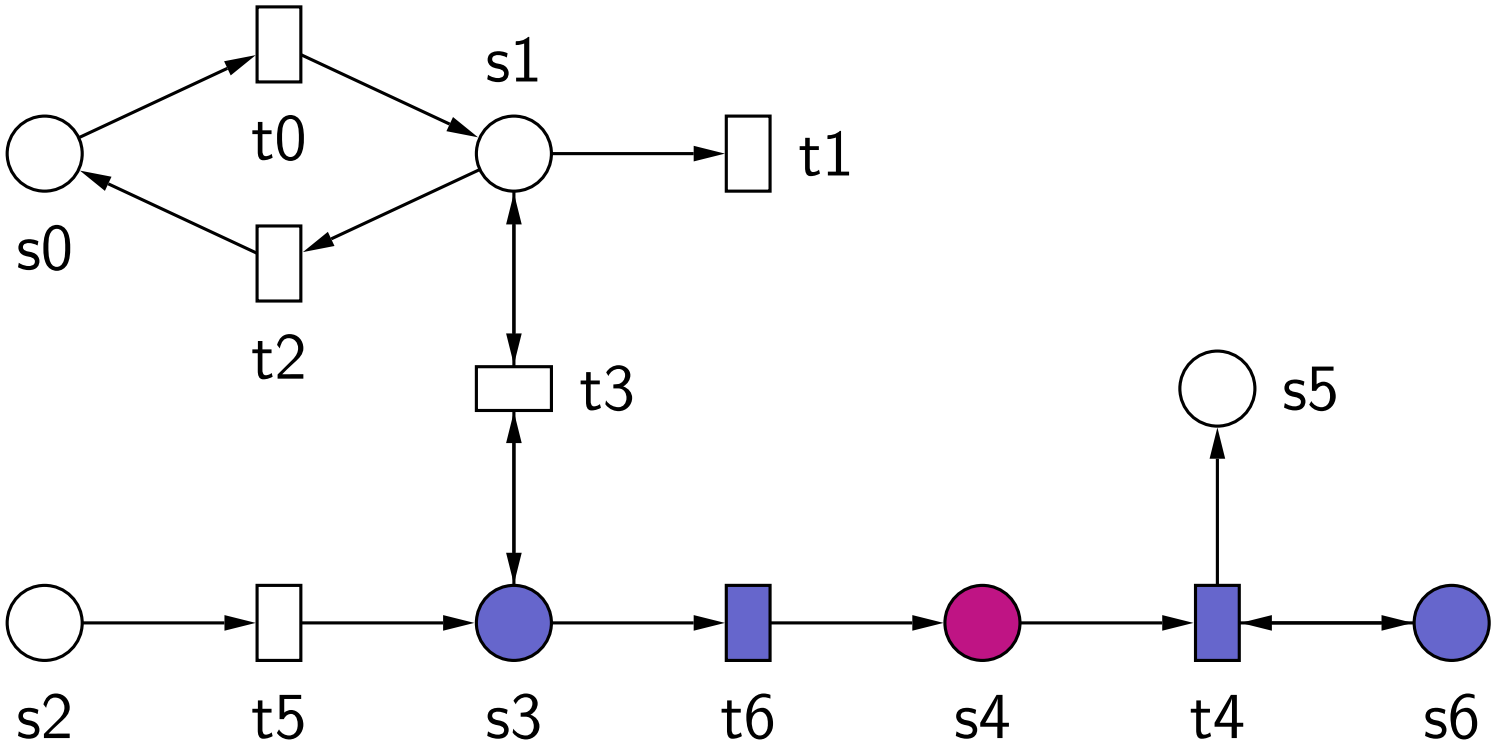
Given: $\Sigma = (S, T, W, M_0)$ and $P \subseteq S$ a non-empty set.

T', $S_{done} := \emptyset$;
$\mathbf{S' := P;}$
while $(\exists s \in (S' \setminus S_{done}))$ {
  while $(\exists \mathbf{t} \in (\mathbf{{}^\bullet s \cup s^\bullet}) : \mathbf{W(s,t) \neq W(t,s)})$ {
    $S' := S' \cup {}^\bullet t$;
    $T' := T' \cup \{t\}$;
  }
  $S_{done} := S_{done} \cup \{s\}$;
}
$W' := W|_{T' \cup S'}$;
$M'_0 := M_0|_{S'}$;
return (S', T', W', M'_0)

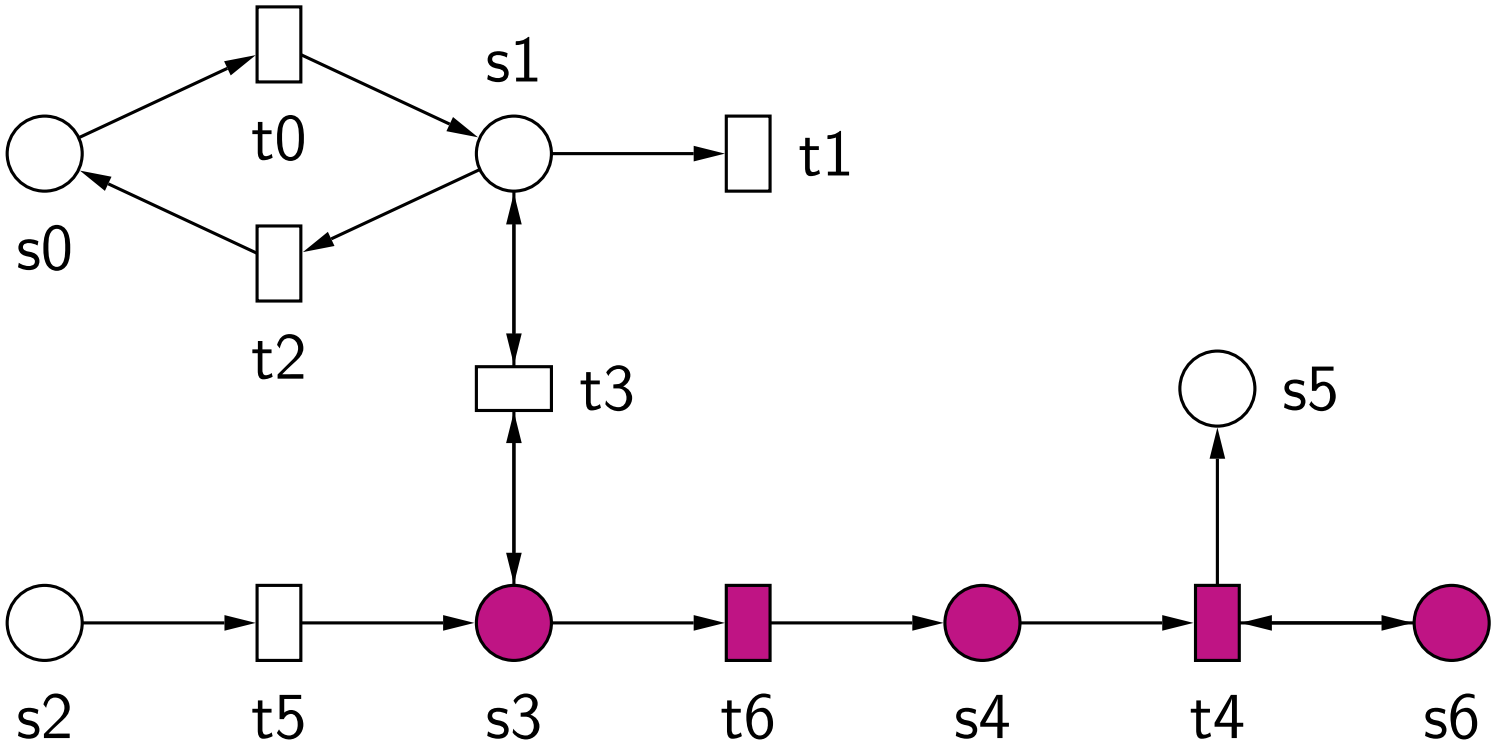
6 Applying the Algorithm



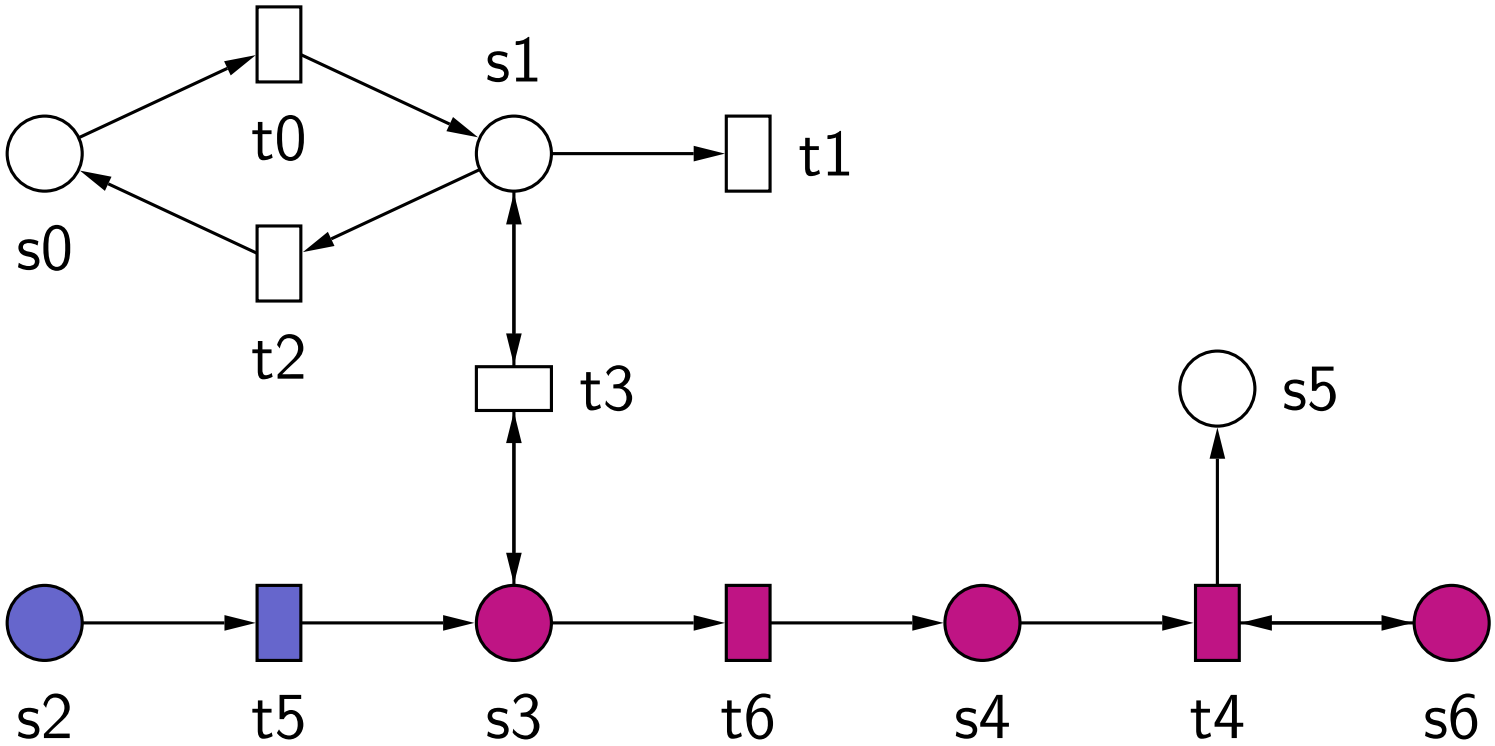
7 Applying the Algorithm



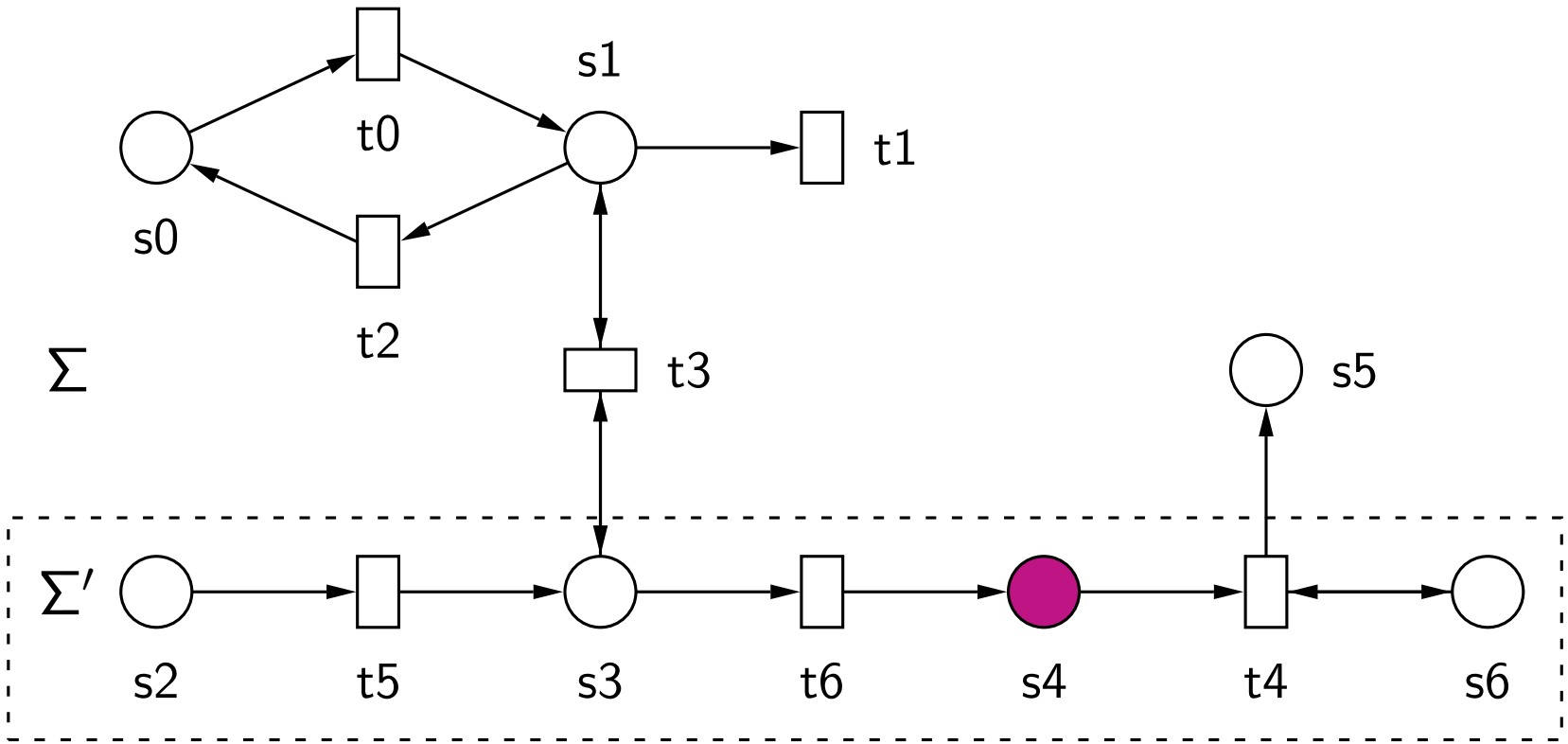
8 Applying the Algorithm



9 Applying the Algorithm



10 Applying the Algorithm



11 Results

Let ϕ be a $\mathbf{CTL}^*_{-\mathbf{x}}$ formula such that ϕ refers to places of the slice Σ'.

$$\mathbf{\Sigma} \models \phi \textbf{ fairly w.r.t } \mathbf{T'} \quad \Leftrightarrow \quad \mathbf{\Sigma'} \models \phi$$

Let ϕ be an **LTL** formula such that ϕ refers to places of the slice Σ'.

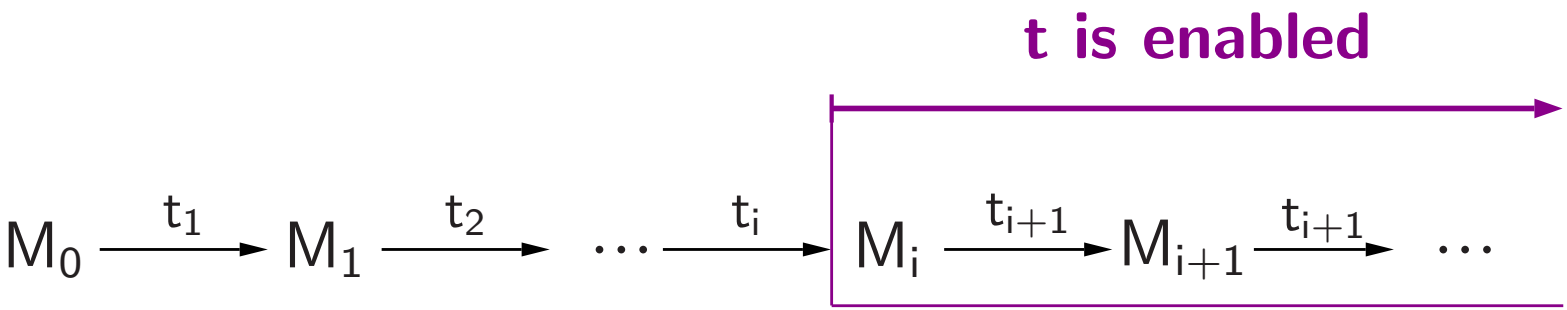
$$\mathbf{\Sigma} \models \phi \quad \Rightarrow \quad \mathbf{\Sigma'} \models \phi$$

12 Fairness

Definition

σ **eventually permanently enables** $t \in T$ iff

- either σ is finite and t is enabled after firing σ
- or σ is infinite and $\exists i, 0 \leq i : \forall j, i \leq j : M_j[t\rangle$

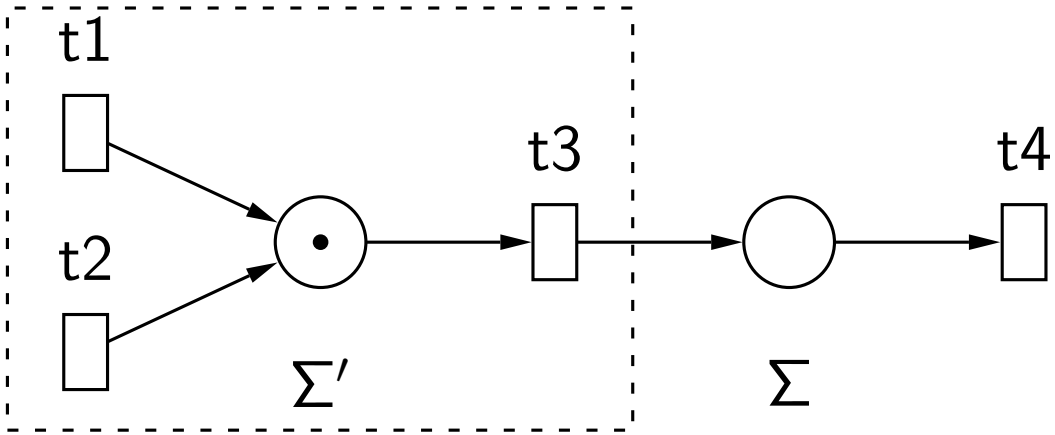


13 Fairness

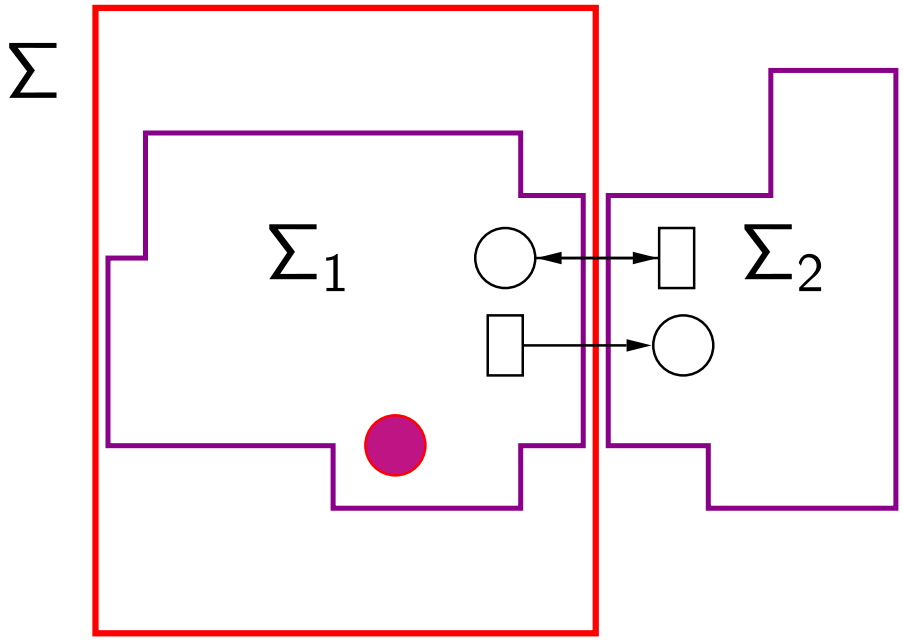
Definition

A **maximal** σ is **fair w.r.t** T' iff

- either σ is finite
- or σ is infinite and if there is a $t \in T'$ it **eventually permanently enables**, it then **fires infinitely often** some $t' \in T'$.



13 Reducible Nets



14 Evaluation

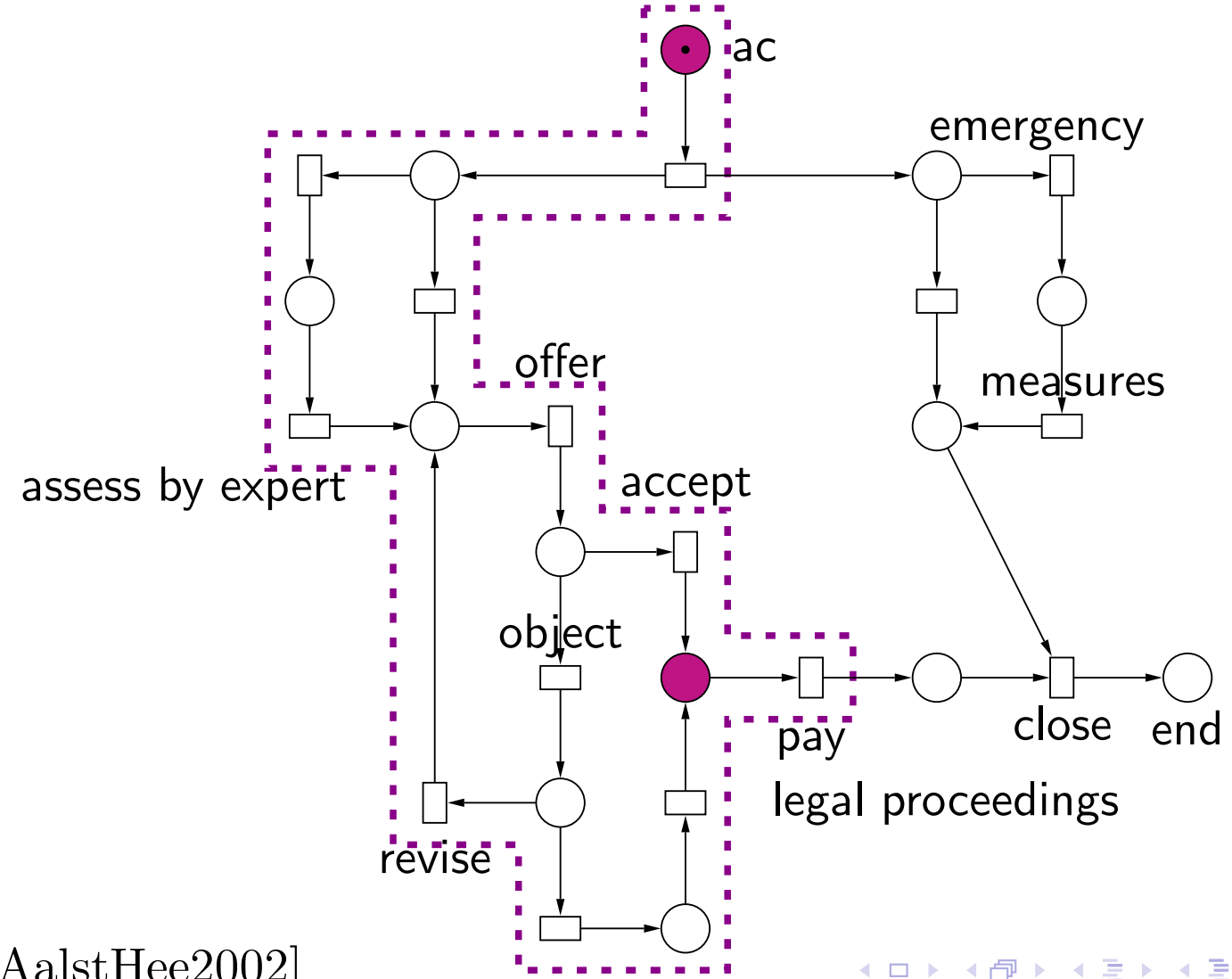
Benchmark set of [Corbett96]

- For each Petri net a slice for each place is built.
- A slice is nice if it has 10% to 85% of the states.
- 6/23 with nice slices
- 67 % of places covered in average

14 Workflows

- A workflow defines the order of **tasks** of a **process**
- Petri nets can represent workflows very intuitively
- Workflow analysis helps to make a process effective and efficient
- A P/T net represents the possible behaviour of a workflow

15 A Workflow Example



[AalstHee2002]

16 Conclusions

- linear-time algorithm
- Petri net is reduced
- state space is reduced

Thanks for listening !

References

Corbett, J.C.: **Evaluating Deadlock Detection Methods for Concurrent Software**. In IEEE Transactions on Software Engineering **22** 3 (1996) 161–180

van der Aalst, Wil; van Hee, Kees: **Workflow Management:Models, Methods and Systems**. The MIT Press (2002)