# Checking Deadlock-Freeness in Petri Nets

Donald Craig
Ph.D. Candidate

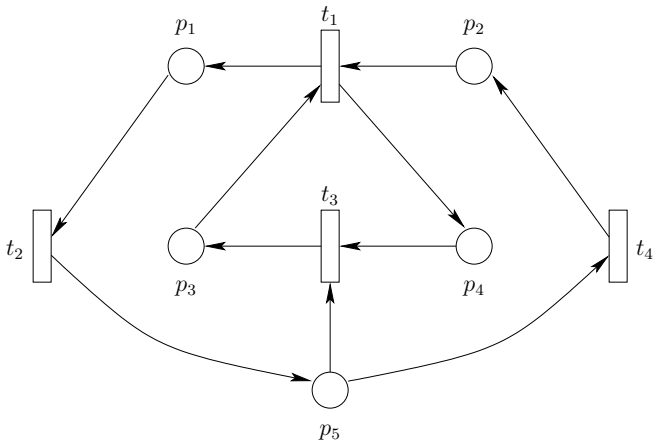
Supervisor:
Dr. Wlodek Zuberek

Department of Computer Science
Memorial University of Newfoundland
March 8, 2006

# Motivation, Purpose and Outline

- Software component interfaces can be represented using Petri nets.
- Multiple component interfaces can be composed using a formal technique (not entirely relevant to this particular presentation).
- In order to assess the compatibility of the resulting composition, we must determine whether the composed net is deadlock free.
- Outline
  - Introduction to Petri nets
  - Reachability analysis
  - Structural analysis
  - Net simplification techniques
    - Similar and essential siphons
    - Parallel paths
    - Alternative paths
  - Conclusions

# Introduction to Petri Nets



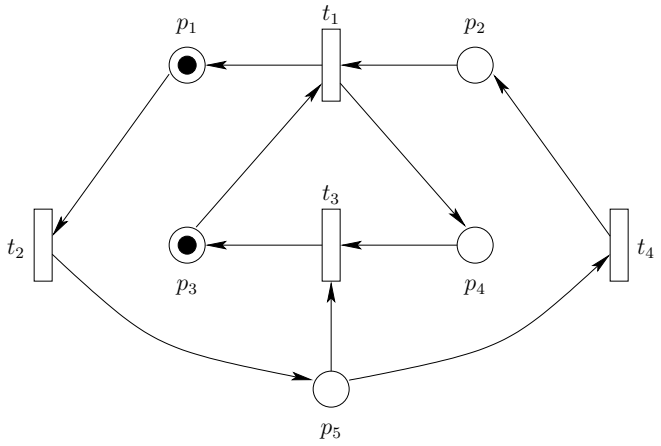

$$\mathbf{C} = \begin{bmatrix} +1 & -1 & 0 & 0 \\ -1 & 0 & 0 & +1 \\ -1 & 0 & +1 & 0 \\ +1 & 0 & -1 & 0 \\ 0 & +1 & -1 & -1 \end{bmatrix}$$

- A Petri net is defined by sets of places (representing conditions) and transitions (representing events) which are connected to each other by directed arcs, $\mathcal{N} = (P, T, A)$. We define:

$$\begin{array}{lll} \mathrm{Inp}(p) = & \{\, t \in T \mid (t,p) \in A \,\}, & \textit{e.g.}\ \mathrm{Inp}(p_5) = \{t_2\} \\ \mathrm{Out}(p) = & \{\, t \in T \mid (p,t) \in A \,\}, & \textit{e.g.}\ \mathrm{Out}(p_5) = \{t_3, t_4\} \\ \mathrm{Inp}(t) = & \{\, p \in P \mid (p,t) \in A \,\}, & \textit{e.g.}\ \mathrm{Inp}(t_1) = \{p_2, p_3\} \\ \mathrm{Out}(t) = & \{\, p \in P \mid (t,p) \in A \,\}, & \textit{e.g.}\ \mathrm{Out}(t_1) = \{p_1\}. \end{array}$$

- The structure of a Petri net can be described by a connectivity matrix $\mathbf{C}$ (the rows correspond to places and the columns correspond to transitions).

# Introduction to Petri Nets (cont'd)

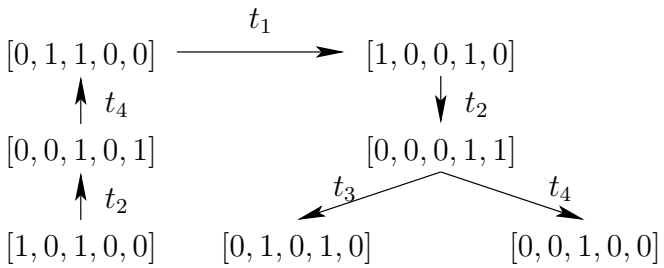



- We can assign *tokens* to the places of a Petri net. This is known as a marked net, $\mathcal{M} = (P, T, A, m_0)$, where $m_0$ is the initial marking function.

- A transition, $t$, is *enabled* in marking $m_0$ iff $\forall p \in \mathrm{Inp}(t) : m_0(p) > 0$. An enabled transition can *fire* — this moves one token from each of the transition's input places to each of its output places.

- When no transition is enabled, the net is *deadlocked*. The *firing sequence* $(t_2, t_4, t_1, t_2, t_3)$ results in the net above becoming deadlock.

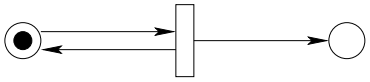
# Checking Deadlock-freeness – Reachability Analysis



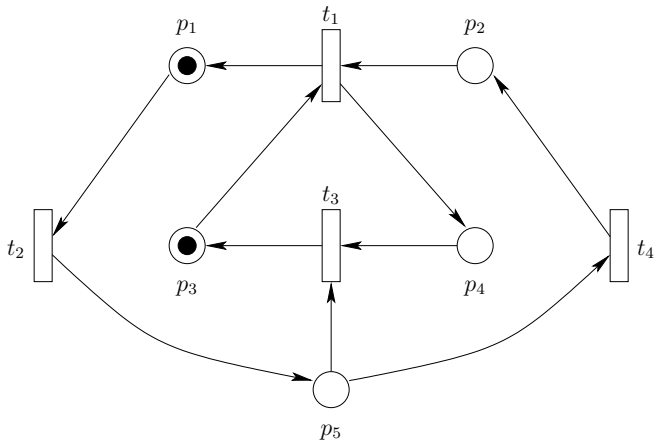

| Node | Marking | Next Marking | Transition |
|---|---|---|---|
| 0 | $[1,0,1,0,0]$ | 1 | $t_2$ |
| 1 | $[0,0,1,0,1]$ | 2 | $t_4$ |
| 2 | $[0,1,1,0,0]$ | 3 | $t_1$ |
| 3 | $[1,0,0,1,0]$ | 4 | $t_2$ |
| 4 | $[0,0,0,1,1]$ | 5 | $t_3$ |
| | | 6 | $t_4$ |
| 5 | $[0,0,1,0,0]$ | dead | $\emptyset$ |
| 6 | $[0,1,0,1,0]$ | dead | $\emptyset$ |

- The *reachability graph* of a marked net can be derived by exhaustively determining all possible markings that are possible (or *reachable*) from the initial marking. The graph may also be expressed in the form of a table.

- Limitations:
  - The number of reachable markings could conceivably be quite large for larger nets. But this may be mitigated using techniques such as Binary Decision Diagrams (BDD); state spaces of up to $10^{100}$ states have been analyzed in this way.
  - Not suitable for unbounded nets: the reachability graph becomes infinite.



# Siphons and Traps




- Siphons and traps are important for the structural analysis of nets.
- A siphon is a subset of places, $S \subseteq P$, such that $\mathrm{Inp}(S) \subseteq \mathrm{Out}(S)$, where: $\mathrm{Inp}(S) = \bigcup_{s \in S} \mathrm{Inp}(s), \mathrm{Out}(S) = \bigcup_{s \in S} \mathrm{Out}(s)$. ***e.g.*** $\{p_1, p_3, p_4\}$ is a siphon since $\{t_1, t_3\} \subseteq \{t_1, t_2, t_3\}$.
- Once a siphon becomes unmarked, it remains unmarked "forever."
- A *minimal siphon* is a siphon which does not include any other siphon. $\{p_3, p_4\}$ is a minimal siphon (consequently $\{p_1, p_3, p_4\}$ is not).
- A trap is a subset of places, $Q \subseteq P$, such that $\mathrm{Out}(Q) \subseteq \mathrm{Inp}(Q)$.

# Checking for Deadlock-freeness – Structural Analysis

[Xie & Chu, 1997] A Petri net is deadlock-free if for each minimal siphon S, either it contains a marked trap or

$$\min \left( \sum_{p \in S} m(p) \right) > 0$$

subject to: $x \geq 0;\ m = m_0 + \mathbf{C}x \geq 0.$

$x$ is the *firing vector* which can be determined through linear programming.

Limitations:

- Determining the objective function requires enumerating all the minimal siphons in a net — for some nets, the number of minimal siphons increases exponentially with the net size.

- If the number of minimal siphons is large, many objective functions may have to be minimized.

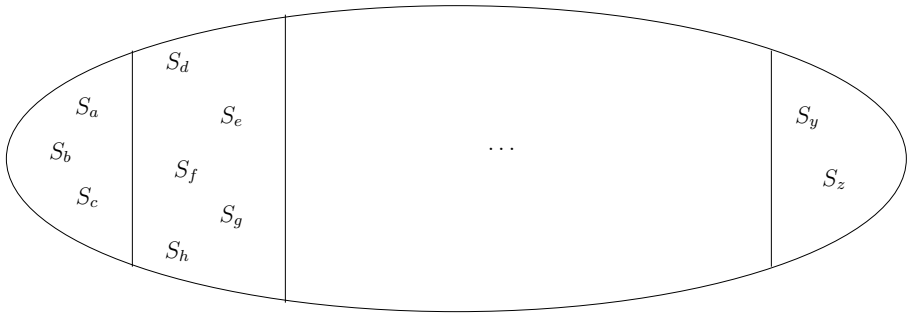
# Similar and Essential Siphons

Definition: Two siphons $S_1$ and $S_2$ in a net $\mathcal{M}$ are similar, $S_1 \sim S_2$, if for all reachable markings, either both are marked or both are unmarked:

$$S_1 \sim S_2 \Leftrightarrow \forall m \in \mathsf{M}(\mathcal{M}) : \mathrm{mark}(S_1, m) = 0 \Leftrightarrow \mathrm{mark}(S_2, m) = 0$$

where $\mathrm{mark}(S, m) = \sum_{p \in S} m(p)$.

Corollary: The relation of siphon similarity is an equivalence relation on the set of siphons of a marked net $\mathcal{M}$, so it implies a partition of this set into classes of similar siphons.




Definition: Set $S = \{S_1, S_2, \ldots, S_n\}$ is a set of essential siphons for $\mathcal{M}$ if no two siphons in $S$ are similar and any other siphon of $\mathcal{M}$ is similar to one of the siphons in $S$.

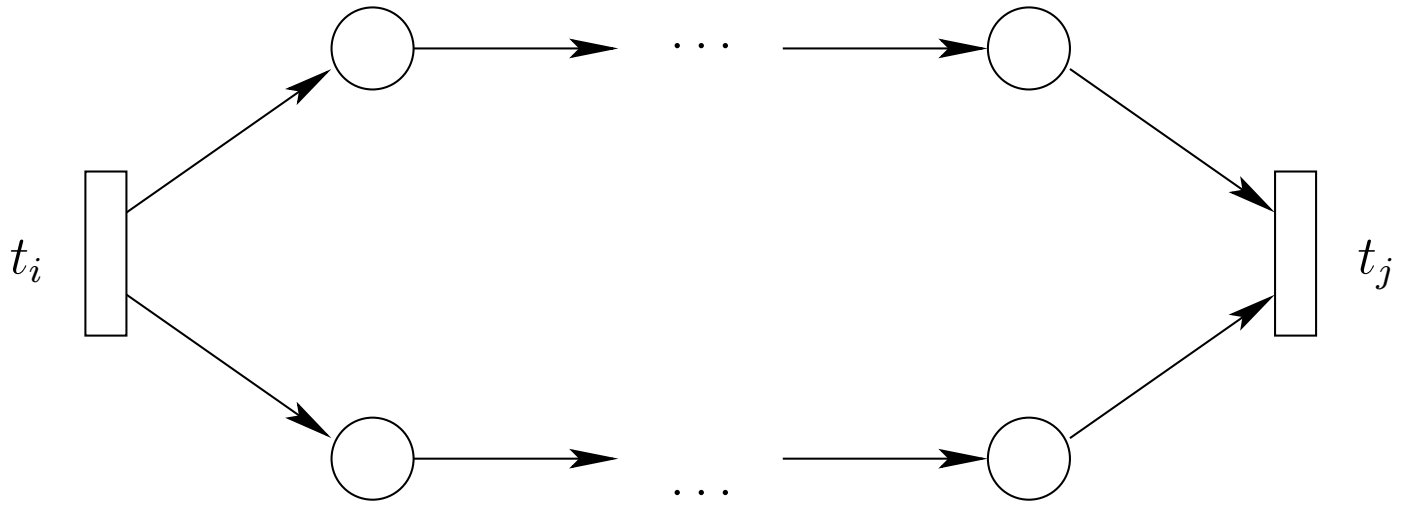
# Reduction of Similar Siphons — Parallel Paths

Definition: A *simple path* in a net $\mathcal{N}$ is a sequence of transitions and places $t_{i_0} p_{i_1} t_{i_1} p_{i_2} \ldots p_{i_k} t_{i_k}$, such that:

$$\begin{array}{ll} (\forall 1 \leq j \leq k & : \mathrm{Inp}(p_{i_j}) = \{t_{i_{j-1}}\} \ \wedge \ \mathrm{Out}(p_{i_j}) = \{t_{i_j}\}) \ \wedge \\ (\forall 1 \leq j < k & : \mathrm{Inp}(t_{i_j}) = \{p_{i_j}\} \ \wedge \ \mathrm{Out}(t_{i_j}) = \{p_{i_{j+1}}\}) \end{array}$$

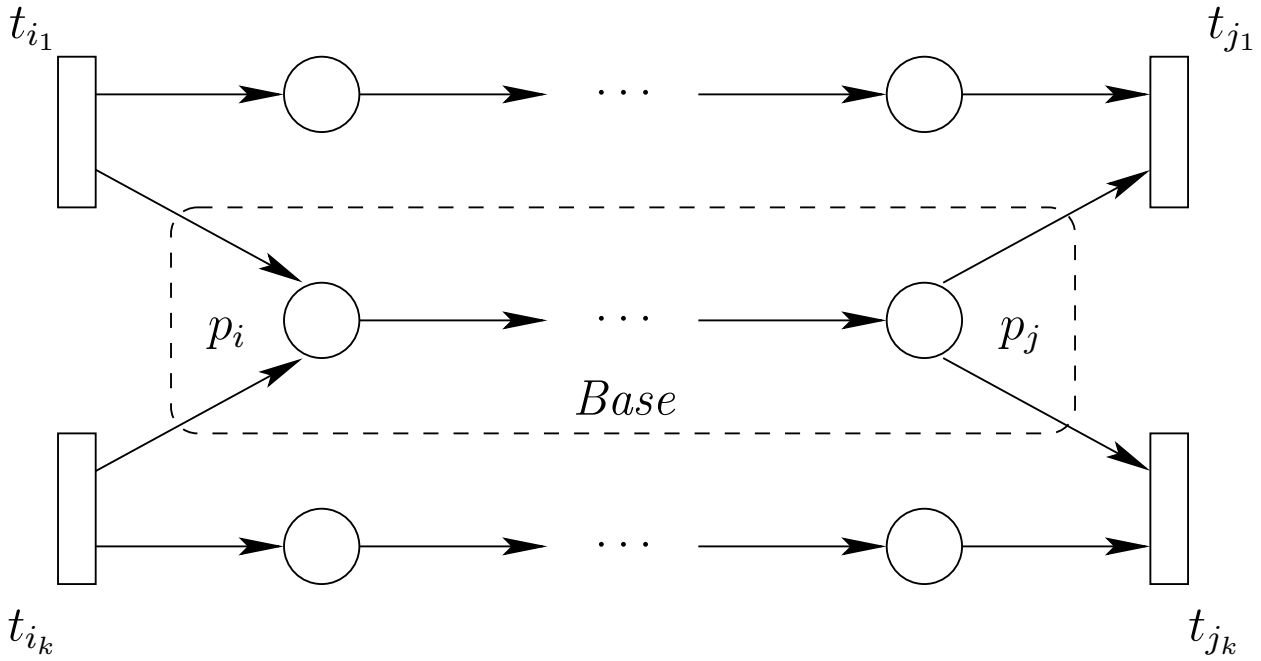
A simple path is denoted by the pair of delimiting transitions, *e.g.* $\mathrm{path}(t_{i_0}, t_{i_k})$.

Definition: *Parallel paths* are any two simple paths which connect the same transitions $t_i$ and $t_j$ (where $t_i = t_{i_0}$ and $t_j = t_{i_k}$).



# Reduction of Similar Siphons — Alternative Paths

Definition: An *alternative path* is a collection of disjoint, simple paths $path(t_{i_1}, t_{j_1})$, $path(t_{i_2}, t_{j_2})$,. . . , $path(t_{i_k}, t_{j_k})$, $t_{i_\ell} \neq t_{i_n}$, $t_{j_\ell} \neq t_{j_n}$, for $1 \leq \ell \leq k$, with an additional common path (called the *base*), connected to all $t_{i_\ell}$, $(t_{i_\ell}, p_i) \in A, 1 \leq \ell \leq k$, and all $t_{j_\ell}$, $(p_j, t_{j_\ell}) \in A$, $\ell = 1, \ldots, k$.




Observe that alternative paths are different from parallel paths because parallel paths are delimited by exactly one pair of transitions. Alternative paths are delimited by more than one pair of transitions and have a shared base.

# Petri Net Simplification/Reduction

- If a net $\mathcal{M}$ has parallel paths, then a set of essential siphons for $\mathcal{M}'$, a net obtained from $\mathcal{M}$ by removing one of the parallel paths, is also a set of essential siphons for $\mathcal{M}$.

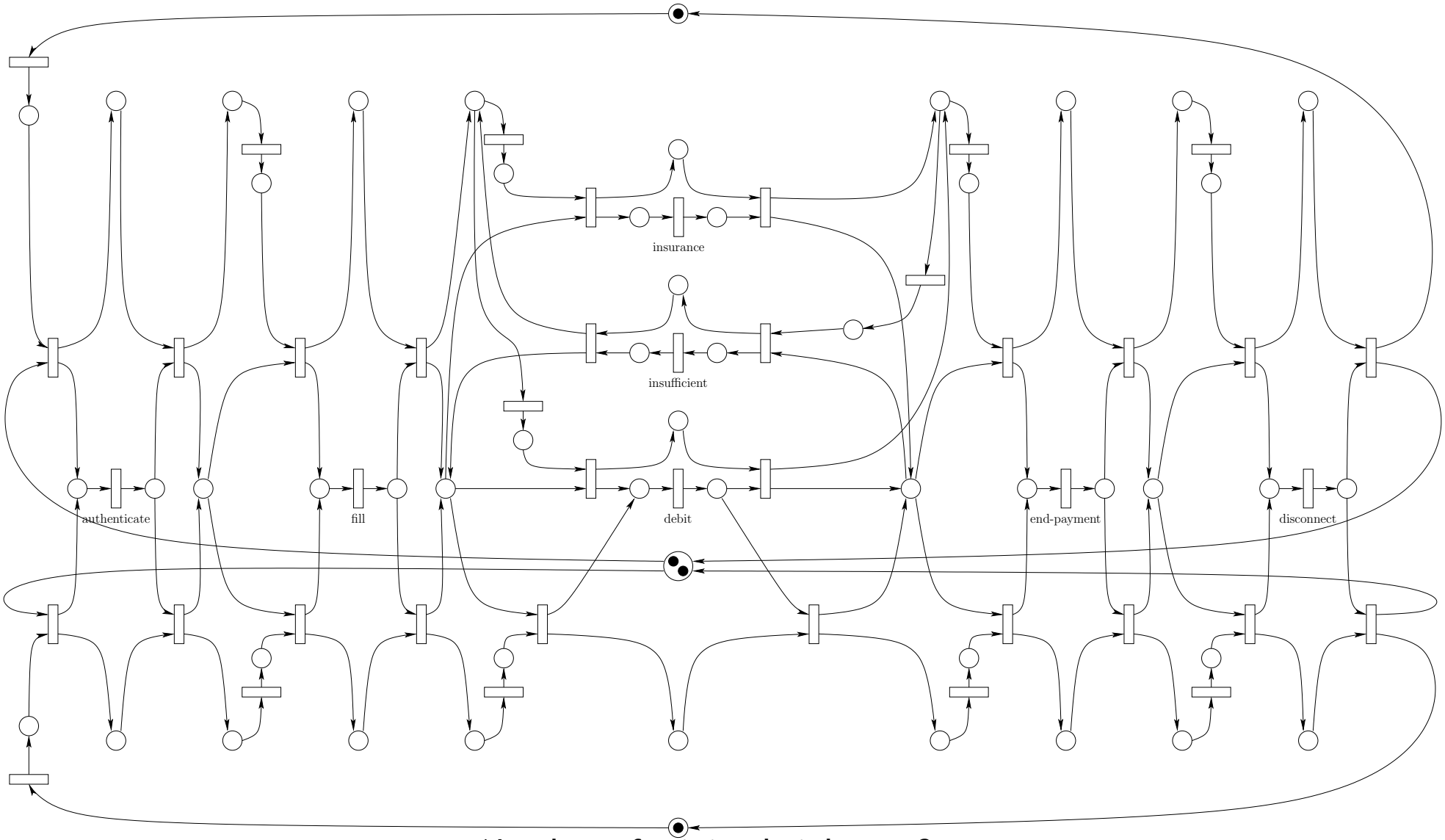
- If a net $\mathcal{M}$ has alternative paths, then a set of essential siphons for $\mathcal{M}'$, a net obtained from $\mathcal{M}$ by removing the base of alternative paths, is also a set of essential siphons for $\mathcal{M}$.

- A set of essential siphons for a net $\mathcal{M}$ can be determined by removing all parallel paths in $\mathcal{M}$ and all bases of alternative paths and finding the siphons in the simplified net $\mathcal{M}'$.

- Any set of essential siphons of $\mathcal{M}$ is sufficient for deadlock analysis of $\mathcal{M}$.

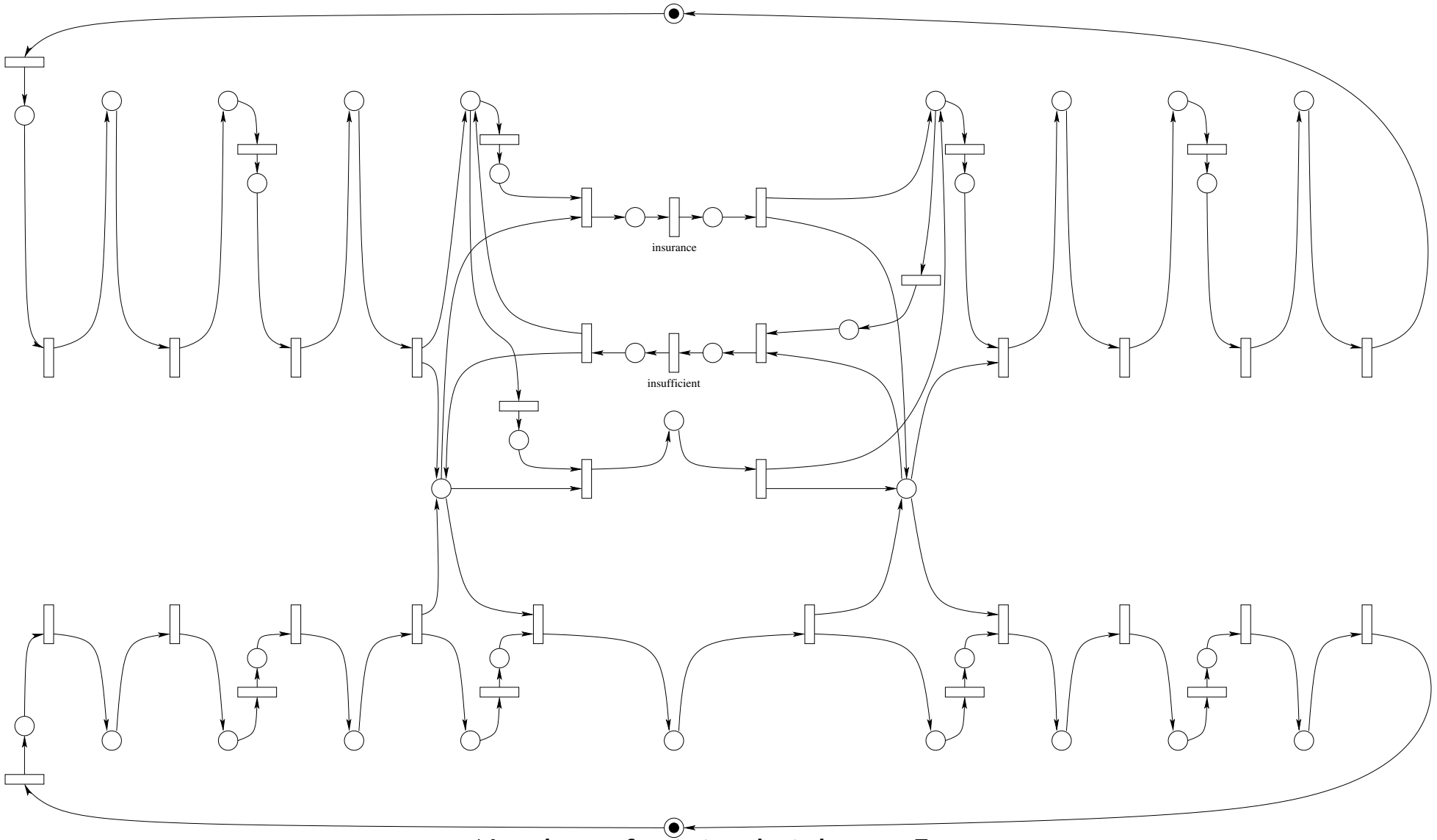
- Removing alternative and parallel paths can make the net significantly easier to analyze without adversely affecting the tests for deadlock-freeness.

# Component Compatibility Example (before)




Number of minimal siphons: ?

# Component Compatibility Example (after)




Number of minimal siphons: 5.

# Concluding Remarks

- Deadlock detection can be done (with varying degrees of success) using reachability analysis (for small and medium size models) or structural analysis (based on minimal siphons) and linear programming.

- In order to make structural analysis more efficient, a net may be reduced, or simplified, while still preserving the relevant properties for deadlock analysis.

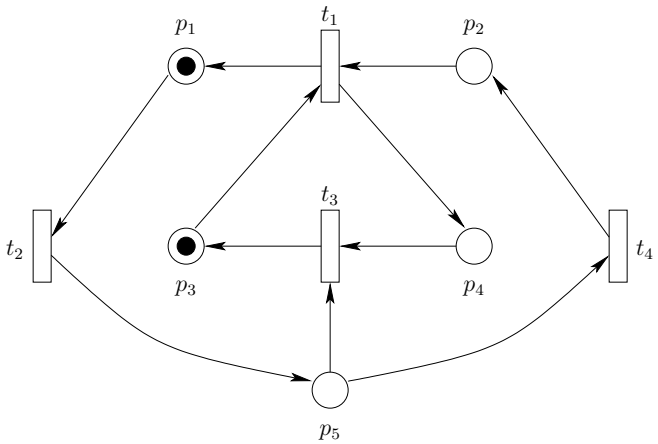
- Two ways to reduce or simplify a net are to remove parallel and alternative paths.

- As mentioned at the beginning, this work has pragmatic considerations: If we represent component interfaces as Petri nets, then we can determine if two or more components are compatible by checking the composed interface for deadlock-freeness.

- Because of the size of the composed models, any simplification of the composed net can have a significant effect on the performance of the verification procedure.

# Supplemental Slides

# Linear Programming example

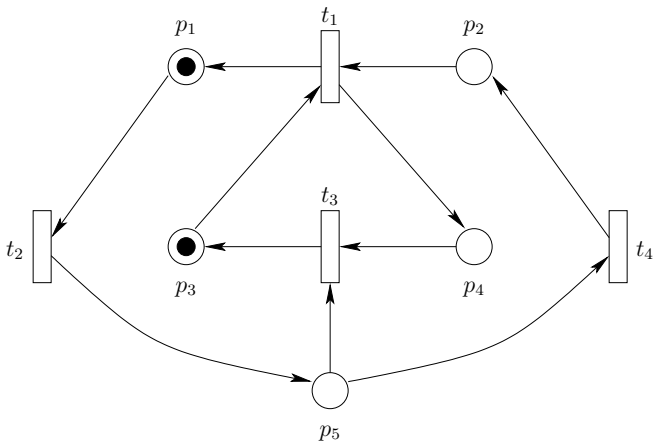



$$\mathbf{C} = \begin{bmatrix} +1 & -1 & 0 & 0 \\ -1 & 0 & 0 & +1 \\ -1 & 0 & +1 & 0 \\ +1 & 0 & -1 & 0 \\ 0 & +1 & -1 & -1 \end{bmatrix}$$

- Minimal siphons: $\{p_1, p_2, p_5\}$, $\{p_1, p_3, p_5\}$ and $\{p_3, p_4\}$.
- But $\{p_3, p_4\}$ is a marked trap and so it cannot become unmarked.
- Considering $S = \{p_1, p_2, p_5\}$, the objective function to be minimized is:

$$\begin{aligned} \sum_{p \in S} m(p) &= 1 + (x_1 - x_2) + (-x_1 + x_4) + (x_2 - x_3 - x_4) \\ &= 1 - x_3 \end{aligned}$$

# Linear Programming example (cont'd)




$$\mathbf{C} = \begin{bmatrix} +1 & -1 & 0 & 0 \\ -1 & 0 & 0 & +1 \\ -1 & 0 & +1 & 0 \\ +1 & 0 & -1 & 0 \\ 0 & +1 & -1 & -1 \end{bmatrix}$$

- The constraints are deduced from the connectivity matrix:

$$\begin{array}{ll} 1 + x_1 - x_2 \geq 0 & -x_1 + x_4 \geq 0 \\ 1 - x_1 + x_3 \geq 0 & x_1 - x_3 \geq 0 \\ x_2 - x_3 - x_4 \geq 0 & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0. \end{array}$$

- Minimizing the tokens in siphon $\{p_1, p_2, p_5\}$ gives the firing vector $[1, 2, 1, 1]$, which can be verified to correspond to the firing sequence $(t_2, t_4, t_1, t_2, t_3)$. Using the same technique on the other siphon, $\{p_1, p_3, p_5\}$, results in the firing vector $[1, 2, 0, 2]$, which corresponds to the firing sequence $(t_2, t_4, t_1, t_2, t_4)$. Both firing sequences result in deadlock.