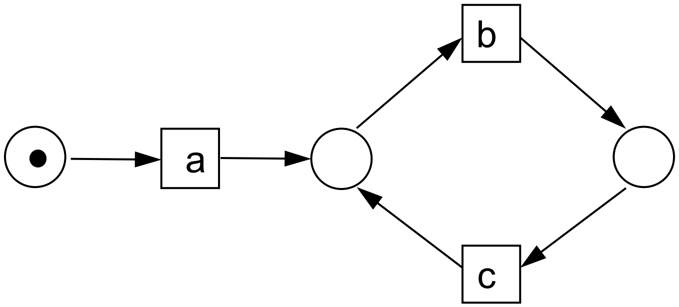
Can a Single Transition Stop an Entire Petri Net?

Jörg Desel
FernUniversität in Hagen
Germany

Can a single transition t stop an entire Petri net?

If transition t does not o…
eventually, then eve…
of the net can occur, i.e., then the net
eventually terminates.

Stopping t causes a shutdown process



a does not stop the net

b and c stop the net

transition t stops its net

for each reachable marking **m**:

m does not enable an infinite occurrence sequence without occurrences of **t**

transition t stops its net

the initial ma… …oes not enable
an in…
with …

Problem of this paper:
How can we decide if t stops its net?

Solution 1:
Solve the LTL-formula
allways eventually t
(can be very inefficient)

the initial marking m_0 does not enable
an infinite occurrence sequence
with only finitely many occurrences of **t**

Solution 2:
Use Petri net analysis techniques!

the initial marking does not enable
an infinite occurrence sequence
with only finitely many occurrences of **t**

How can we decide if a transition t stops its net?

1st case: the net is bounded

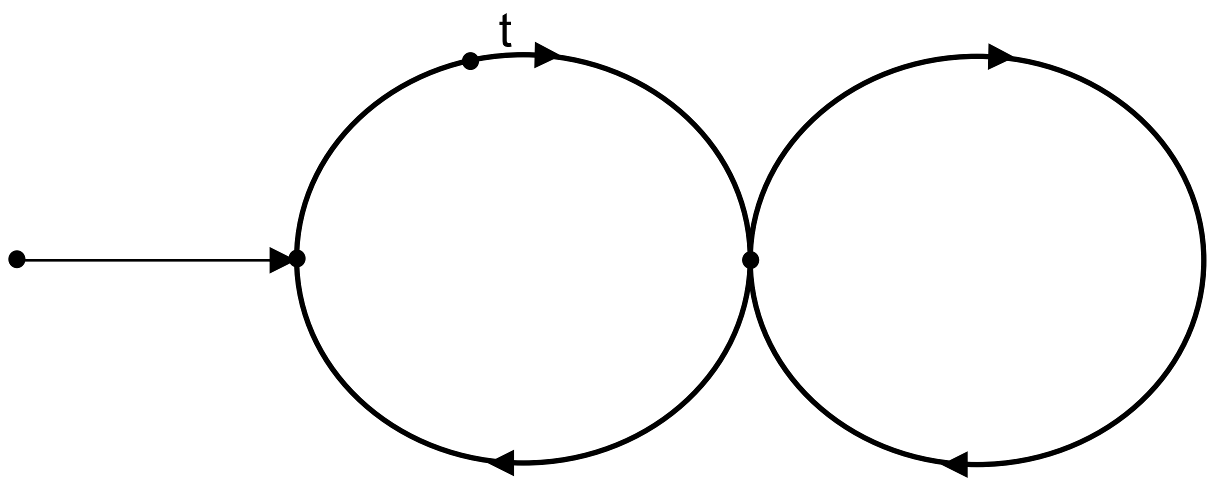
(simple) Theorem:

transition t stops the (bounded) net if and only if

each cycle of the reachability graph contains an edge labeled by t

How can we decide if a transition t stops its net?

1st case: the net is bounded



How can we decide if a transition t stops its net?

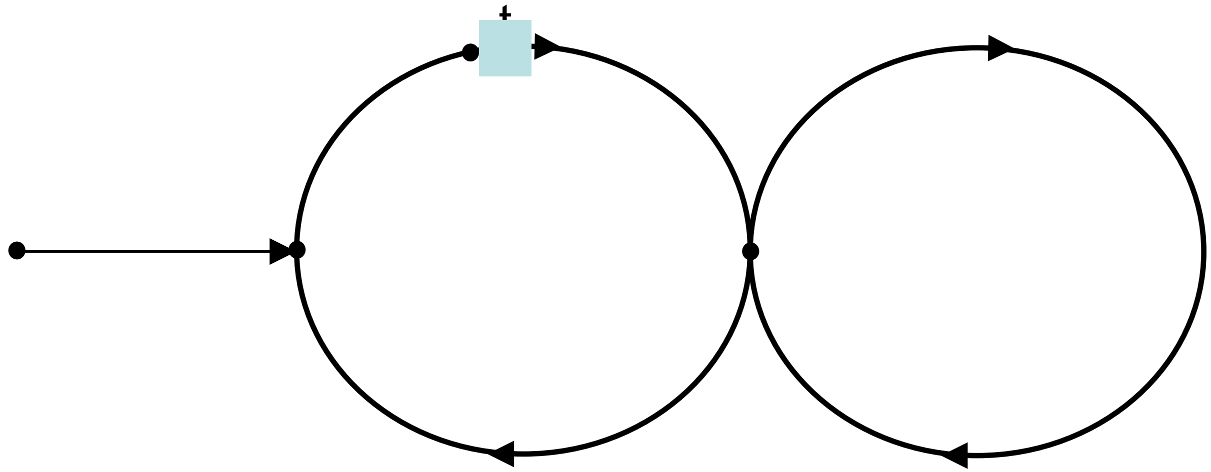
1st case: the net is bounded

Algorithm:

- construct the reachability graph
- delete all edges labelled by t
- check if the remaining graph (which is not necessarily connected) has a cycle

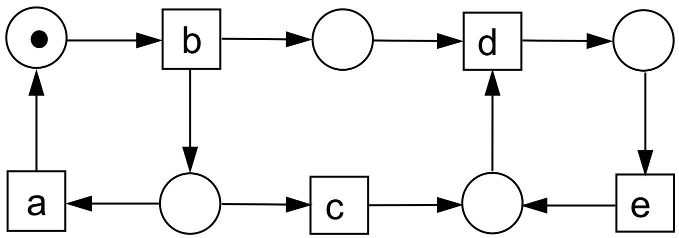
How can we decide if a transition t stops its net?

1st case: the net is bounded



How can we decide if a transition t stops its net?

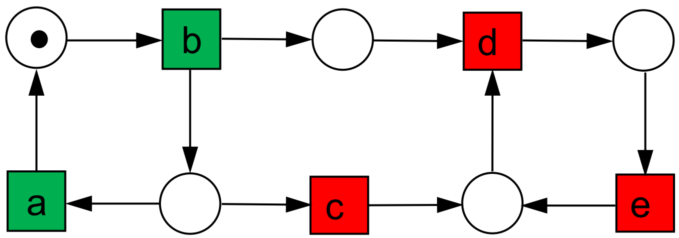
2nd case: the net is **un**bounded



which transitions stop the net?

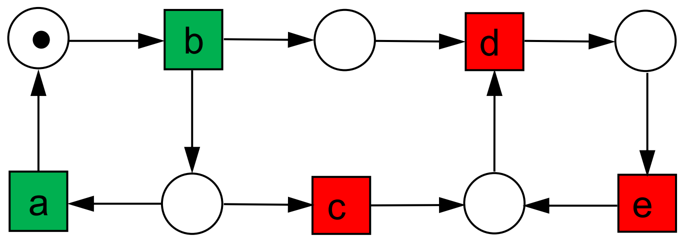
How can we decide if a transition t stops its net?

2nd case: the net is **un**bounded



which transitions stop the net?
a and b

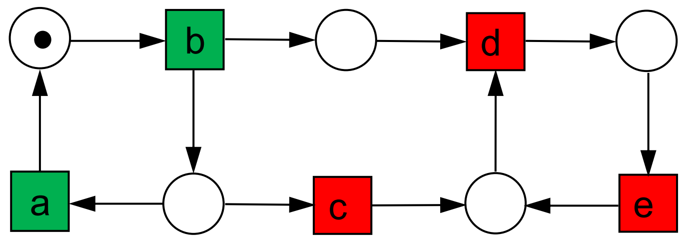
How can we decide if a transition t stops its net?

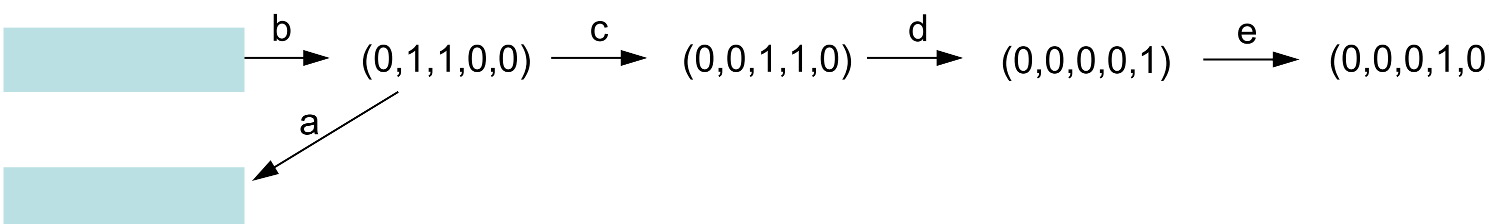


(1,0,0,0,0) $\xrightarrow{b}$ (0,1,1,0,0) $\xrightarrow{c}$ (0,0,1,1,0) $\xrightarrow{d}$ (0,0,0,0,1) $\xrightarrow{e}$ (0,0,0,1,0)

coverability graph

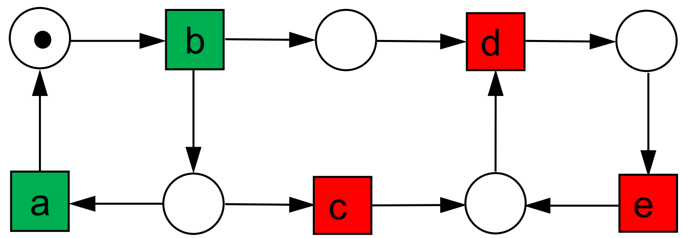
How can we decide if a transition t stops its net?

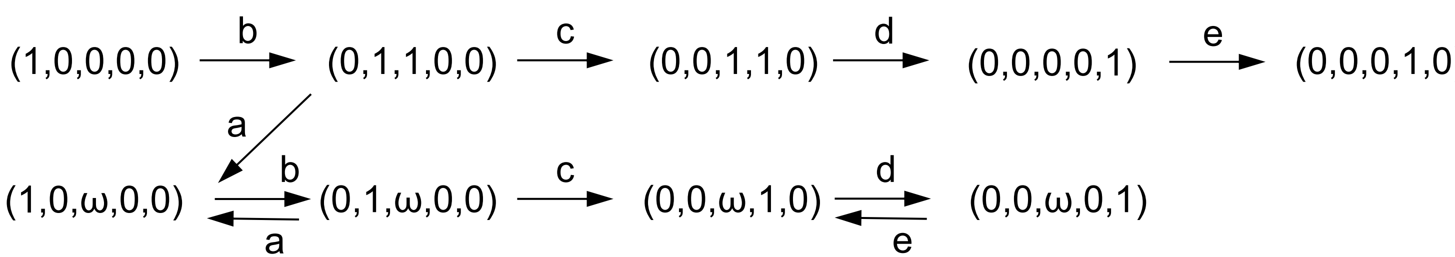




coverability graph

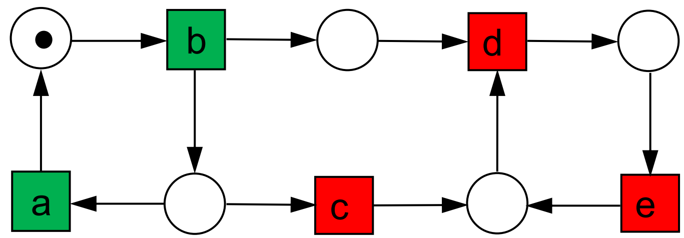
How can we decide if a transition t stops its net?

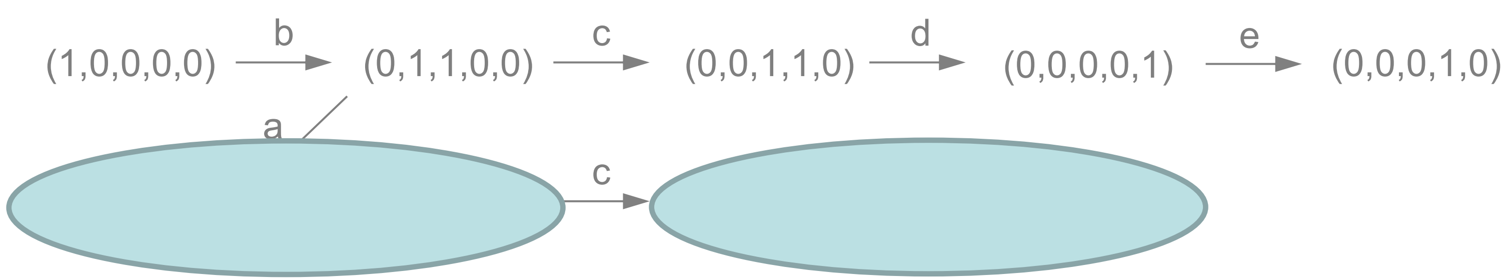




coverability graph

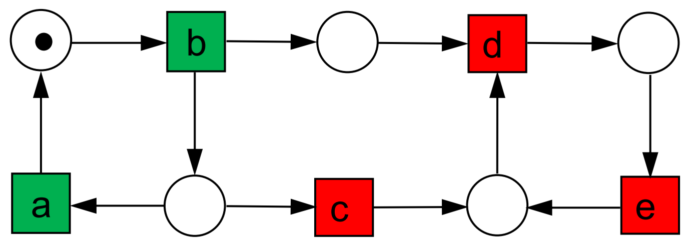
How can we decide if a transition t stops its net?

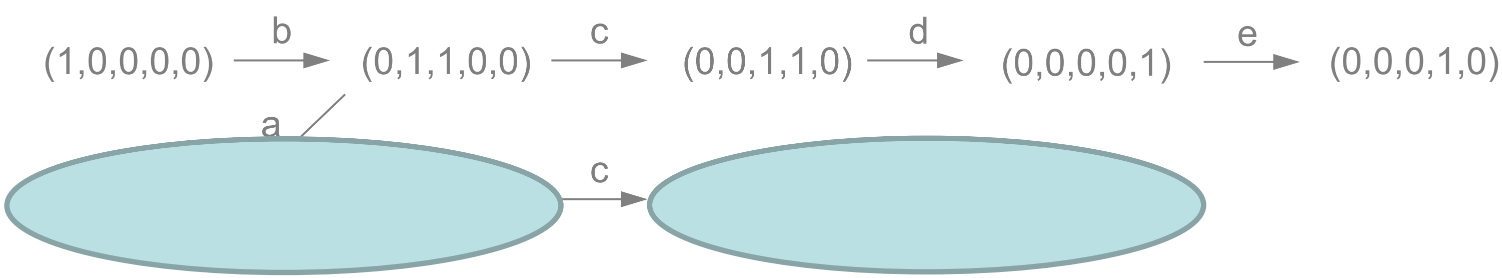




two cycles

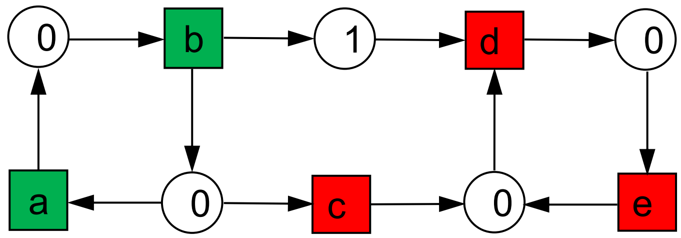
How can we decide if a transition t stops its net?





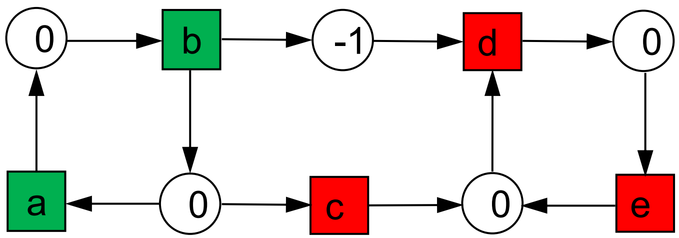
two cycles – no help (they do not distinguish a/b and c/d/e)

How can we decide if a transition t stops its net?

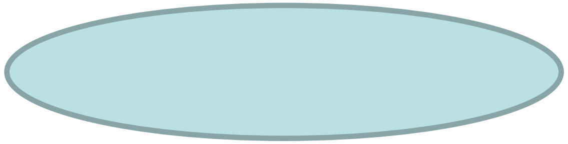


effect of a cycle

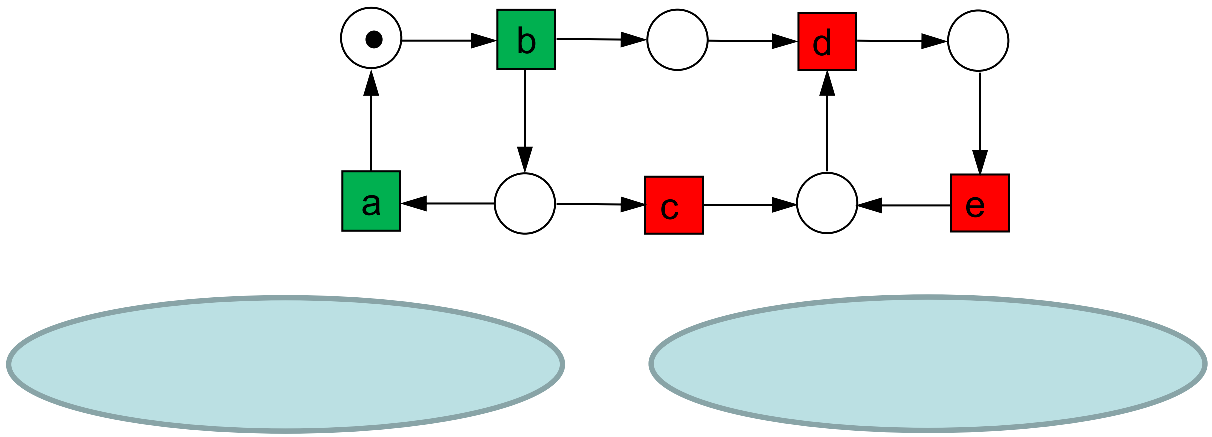
How can we decide if a transition **t** stops its net?



effect of a cycle

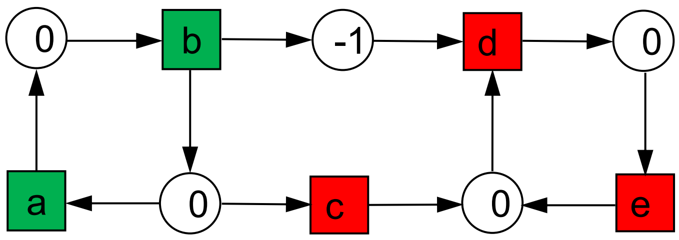


How can we decide if a transition t stops its net?



- the effect of a cycle is 0 for non-ω-marked places (and hence 0 everywhere for bounded nets)
- for ω-marked places, the effect of a cycle can be negative, 0, or positive

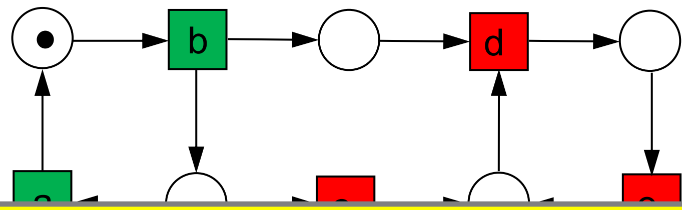
How can we decide if a transition t stops its net?



- cycles with negative effect on a place cannot cycle infinitely (**decreasing cycle**)

- cycles without negative effect on a place can cycle infinitely (**non-decreasing cycle**)

How can we decide if a transition t stops its net?



Theorem:

A transition t stops its net if and only if

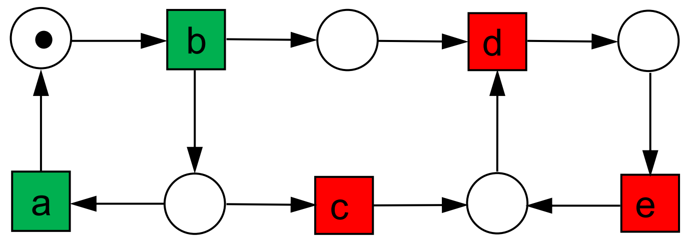
each non-decreasing cycle of the coverability graph contains an edge

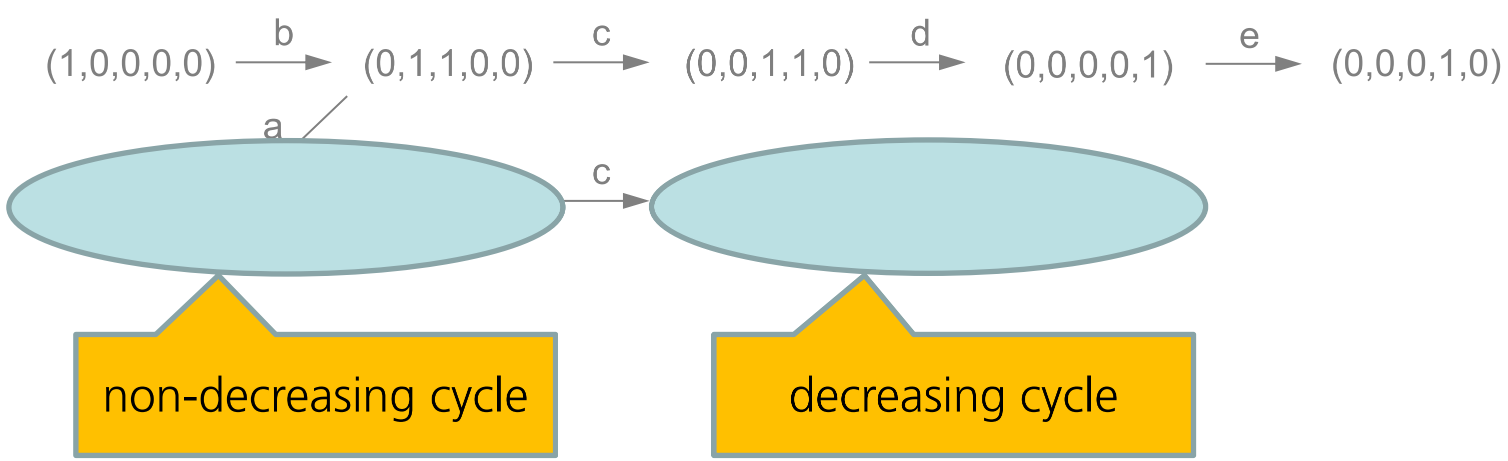
labeled by t

non-decreasing cycle

decreasing cycle

How can we decide if a transition t stops its net?





How can we decide if a transition t stops its net?

Theorem (bounded case):

A transition t stops its net if and only if

each cycle of the reachability graph contains an edge labeled by t

Theorem:

A transition t stops its net if and only if

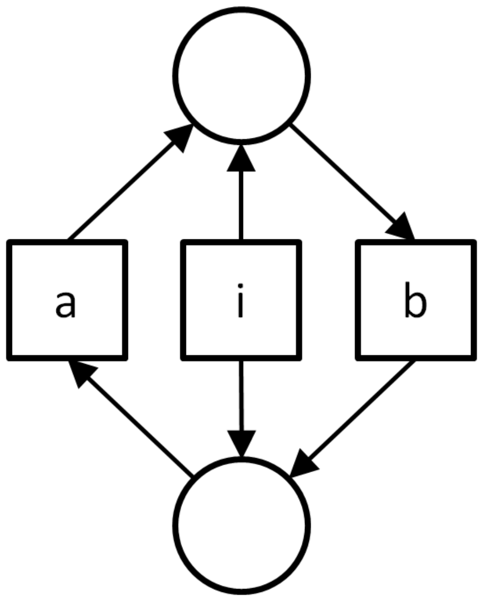
each non-decreasing cycle of the coverability graph contains an edge labeled by t

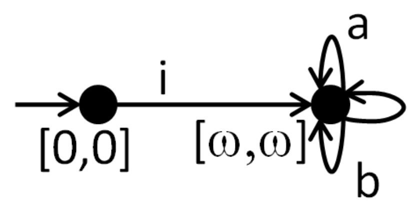
How can we decide if a transition t stops its net

Problem:
there are infinitely many cycles
(not only elementary cycles)

Algorithm

- Construct the coverability graph
- Check if each non-decreasing cycle contains an edge labeled by t
  (if the net is bounded, then
  - coverability graph = reachability graph
  - all cycles are non-decreasing)

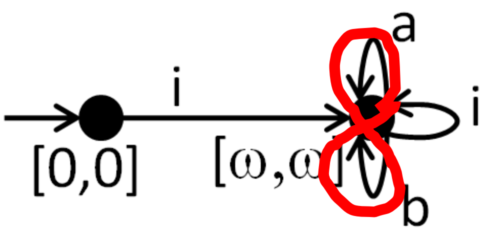




Does tra

So we have to consider arbitrary cycles (closed paths), not only elementary cycles

- the cy
- the cy
- the cycle **b** is n n-decreasing
- the cycle **ab** is non-decreasing and does not contain **i**



Properties of *relevant* cycles (closed paths) **π:**

(1) The subgraph generated by π is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π

(3) All vertices of π have the same **ω**-marked places

(4) For each **ω**-marked place **s**: $\# u \in \bullet s$ in π $\geq$ $\# u \in s\bullet$ in π

(5) No arc in π is marked by transition **t** (in the example: **i**)

Properties of relevant cycles (closed paths) **π:**

(1) The subgraph generated by π is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π

(3) All vertices of π have the same **ω**-marked places

(4) For each **ω**-marked place **s**: # u $\in$ •**s** in π $\geq$ # u $\in$ •**s** in π

(5) No arc in π is marked by transition **t**

Let $(a_1, a_2, a_3, \ldots.)$ denote the arcs of the coverability graph

Properties of relevant cycles (closed paths) π:

Properties of relevant multisets $(x_1, x_2, x_3, \ldots.)$ of arcs:

The multiset is actually $x_1 a_1 + x_2 a_2 + \ldots$

(1) The subgraph generated by π is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π

(3) All vertices of π have the same $\boldsymbol{\omega}$-marked places

(4) For each $\boldsymbol{\omega}$-marked place **s**: # $u \in \bullet \mathbf{s}$ in $\pi \geq$ # $u \in \bullet \mathbf{s}$ in π

(5) No arc in π is marked by transition **t**

Let $(a_1, a_2, a_3, \ldots.)$ denote the arcs of the coverability graph.

Properties of relevant cycles (closed paths) **π:**
Properties of relevant multisets $(x_1, x_2, x_3, \ldots.)$ of arcs:

(1) The subgraph generated by π is strongly connected
(1) The subgraph generated by $(x_1, x_2, x_3, \ldots.)$ is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π

(3) All vertices of π have the same **ω**-marked places

(4) For each **ω**-marked place **s**: $\# u \in \bullet s$ in $\pi \geq \# u \in \bullet s$ in π

(5) No arc in π is marked by **t**

Let (a_1, a_2, a_3, ….) denote the arcs of the coverability graph.

Properties of relevant cycles (closed paths) **π:**
Properties of relevant multisets (x_1, x_2, x_3, ….) of arcs:

(1) The subgraph generated by π is strongly connected
(1) The subgraph generated by (x_1, x_2, x_3, ….) is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π
(2) For each vertex v: $\sum_{a_i \in in(v)} x_i = \sum_{a_i \in out(v)} x_i$

(3) All vertices of π have the same **ω**-marked places

(4) For each **ω**-marked place **s**: # u $\in$ •**s** in π $\geq$ # u $\in$ •**s** in π

(5) No arc in π is marked by **t**

Let $(a_1, a_2, a_3, \ldots.)$ denote the arcs of the coverability graph.

Properties of relevant cycles (closed paths) π:

Properties of relevant multisets $(x_1, x_2, x_3, \ldots.)$ of arcs:

(1) The subgraph generated by π is strongly connected

(1) The subgraph generated by $(x_1, x_2, x_3, \ldots.)$ is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π

(2) For each vertex v: $\sum_{a_i \in in(v)} x_i = \sum_{a_i \in out(v)} x_i$

(3) All vertices of π have the same **ω**-marked places

(3) If an arc a_i connects vertices with different **ω**-marked places then $x_i = 0$

(4) For each **ω**-marked place **s**: # u $\in$ •**s** in π $\geq$ # u $\in$ •**s** in π

(5) No arc in π is marked by **t**

Let (a_1, a_2, a_3, ….) denote the arcs of the coverability graph.

Properties of relevant cycles (closed paths) **π:**
Properties of relevant multisets (x_1, x_2, x_3, ….) of arcs:

(1) The subgraph generated by π is strongly connected
(1) The subgraph generated by (x_1, x_2, x_3, ….) is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π
(2) For each vertex v: $\sum_{a_i \in in(v)} x_i = \sum_{a_i \in out(v)} x_i$

(3) All vertices of π have the same **ω**-marked places
(3) If an arc a_i connects vertices with different **ω**-marked places then $x_i = 0$

(4) For each **ω**-marked place **s**: # u ∈ •**s** in π ≥ # u ∈ •**s** in π
(4) For each **ω**-marked place **s**: $\sum_{u \in {}^\bullet s} \sum_{\lambda(a_i)=u} x_i \geq \sum_{u \in s^\bullet} \sum_{\lambda(a_i)=u} x_i$

(5) No arc in π is marked by **t**

Let $(a_1, a_2, a_3, \ldots.)$ denote the arcs of the coverability graph.

Properties of relevant cycles (closed paths) π:

Properties of relevant multisets $(x_1, x_2, x_3, \ldots.)$ of arcs:

(1) The subgraph generated by π is strongly connected

(1) The subgraph generated by $(x_1, x_2, x_3, \ldots.)$ is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π

(2) For each vertex v: $\sum_{a_i \in in(v)} x_i = \sum_{a_i \in out(v)} x_i$

(3) All vertices of π have the same **ω**-marked places

(3) If an arc a_i connects vertices with different **ω**-marked places then $x_i = 0$

(4) For each **ω**-marked place **s**: # u ∈ •**s** in π ≥ # u ∈ •**s** in π

(4) For each **ω**-marked place **s**: $\sum_{u \in {}^\bullet s} \sum_{\lambda(a_i)=u} x_i \geq \sum_{u \in s^\bullet} \sum_{\lambda(a_i)=u} x_i$

(5) No arc in π is marked by **t**

(5) If a_i is labeled by **t** then $x_i = 0$

Let $(a_1, a_2, a_3, \ldots)$ denote the arcs of the coverability graph.

Properties of relevant cycles (closed paths) π:
Properties of relevant multisets $(x_1, x_2, x_3, \ldots)$ of arcs:

(1) The subgraph generated by π is strongly connected
(1) The subgraph generated by $(x_1, x_2, x_3, \ldots)$ is strongly connected

(2) For each vertex: # ingoing arcs in π = # outgoing arcs in π
(2) For each vertex v: $\sum_{a_i \in in(v)} x_i = \sum_{a_i \in out(v)} x_i$

(3) All vertices of π have the same **ω**-marked places
(3) If an arc a_i connects vertices with different **ω**-marked places then $x_i = 0$

(4) For each **ω**-marked place **s**: # u ∈ •**s** in π ≥ # u ∈ •**s** in π
(4) For each **ω**-marked place **s**: $\sum_{u \in {}^\bullet s} \sum_{\lambda(a_i)=u} x_i \geq \sum_{u \in s^\bullet} \sum_{\lambda(a_i)=u} x_i$

(5) No arc in π is marked by **t**
(5) If a_i is labeled by **t** then $x_i = 0$

(6) $x_1, x_2, x_3, \ldots \geq 0$

Let (a_1, a_2, a_3, ….) denote the arcs of the coverability graph.

Properties of relevant multisets (x_1, x_2, x_3, ….) of arcs:

(1) The subgraph generated by (x_1, x_2, x_3, ….) is strongly connected

Theorem:

A multiset **(x_1, x_2, x_3, ….)** of arcs satisfies (1) to (6)

if and only if

there is a *relevant* closed path π such that

x_i denotes the number of occurrences of arc **a_i** in π

(2) Fo

(3) If a … hen $x_i = 0$

(4) For

$u \in {}^\bullet s \quad \lambda(a_i)=u \qquad u \in s^\bullet \quad \lambda(a_i)=u$

(5) If a_i is labeled by **t** then $x_i = 0$

(6) x_1, x_2, x_3, …. ≥ 0

Let (a_1, a_2, a_3, …

Properties of rele

(1) The subgraph generat… a_3, ….) is strongly connected

Problem:
there are infinitely many solutions

Algorithm:

For each solution **(s_1, s_2, s_3, ….)** of the Linear Program check whether the subgraph generated by **(s_1, s_2, s_3, ….)** is strongly connected.

If we find such a solution, then **t** does not stop ist net

A Linear Program

Let (a_1, a_2, a

Properties o

(1) The subgraph gene ... x_3, ….) is strongly connected

Each solution is a linear combination of base solutions
(and there are only finitely many base solutions)

Algorithm:

For each linear combination of base solutions **(s_1, s_2, s_3, ….)** of the Linear Program check whether the generated subgraph generated by **(s_1, s_2, s_3, ….)** is strongly connected.

If we find such linear combination, then **t** does not stop ist net

A Linear Program

Let (

Replace arbitrary coefficients > 0 by 1
(the generated subgraph remains the same)

Prop

(1) The subgraph ... x_2, x_3, ...) is strongly connected

Algorithm

For each sum of base solutions **(s_1, s_2, s_3, ...)**

of the Linear ... check whether the generated subgraph

ge

If w

So we consider all nonempty subsets
of a set of base solutions

A Linear Program

In the paper (final section):

Algorithm:

Instead of inspecting all (exponentially many) subsets of base solutions, the algorithm runs in linear time in the number of base solutions and finds a subset that generates a strongly connected subgraph (if such a subgraph exists)

Recursive Algorithm:

For a set of base solutions **S**

- construct the subgraph of the coverability graph
- If this graph is strongly connected then stop (**t** does not stop its net)
- for each strongly connected component of the subgraph:
    - consider the maximal subset **S‘** $\subseteq$ **S** such that the subgraph generated by **S‘** is within the strongly connected component
    - If **S‘** is not empty then call this algorithm for **S‘**

Initially, call the algorithm for the set of all base solutions.

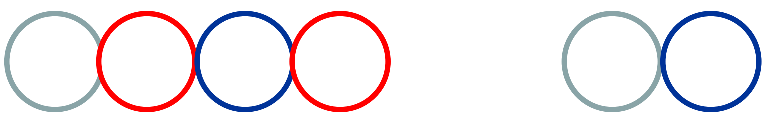
If it comes to ist proper end, then stop (**t** stops its net)

For a set of base solutions **S**

- construct the subgraph of the coverability graph
- If this graph is strongly connected then stop (**t** does not stop its net)
- for each strongly connected component of the subgraph:
    - consider the maximal subset **S‘** $\subseteq$ **S** such that the subgraph generated by **S‘** is within the strongly connected component
    - If **S‘** is not empty then call this algorithm for **S‘**

Initially, call the algorithm for the set of all base solutions.

If it comes to ist proper end, then stop (**t** stops its net)

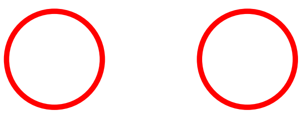


For a set of base solutions **S**

- construct the subgraph of the coverability graph
- If this graph is strongly connected then stop (**t** does not stop its net)
- for each strongly connected component of the subgraph:
    - consider the maximal subset **S‘** $\subseteq$ **S** such that the subgraph generated by **S‘** is within the strongly connected component
    - If **S‘** is not empty then call this algorithm for **S‘**

Initially, call the algorithm for the set of all base solutions.

If it comes to ist proper end, then stop (**t** stops its net)



For a set of base solutions **S**

- construct the subgraph of the coverability graph
- If this graph is strongly connected then stop (**t** does not stop its net)
- for each strongly connected component of the subgraph:
    - consider the maximal subset **S'** $\subseteq$ **S** such that the subgraph generated by **S'** is within the strongly connected component
    - If **S'** is not empty then call this algorithm for **S'**

Initially, call the algorithm for the set of all base solutions.

If it comes to ist proper end, then stop (**t** stops its net)

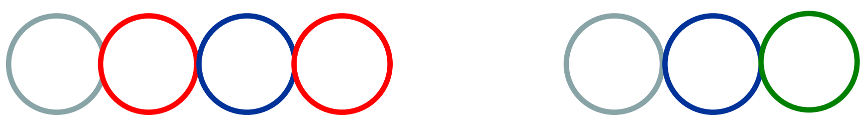
S' empty $\Rightarrow$ proper end (t stops its net)

For a set of base solutions **S**

- construct the subgraph of the coverability graph
- If this graph is strongly connected then stop (**t** does not stop its net)
- for each strongly connected component of the subgraph:
  - consider the maximal subset **S‘** $\subseteq$ **S** such that the subgraph generated by **S‘** is within the strongly connected component
  - If **S‘** is not empty then call this algorithm for **S‘**

Initially, call the algorithm for the set of all base solutions.

If it comes to ist proper end, then stop (**t** stops its net)



For a set of base solutions **S**

- construct the subgraph of the coverability graph
- If this graph is strongly connected then stop (**t** does not stop its net)
- for each strongly connected component of the subgraph:
  - consider the maximal subset **S‘** $\subseteq$ **S** such that the subgraph generated by **S‘** is within the strongly connected component
  - If **S‘** is not empty then call this algorithm for **S‘**

Initially, call the algorithm for the set of all base solutions.

If it comes to ist proper end, then stop (**t** stops its net)

Prof. Dr. Jörg Desel

For a set of base solutions **S**

- construct the subgraph of the coverability graph
- If this graph is strongly connected then stop (**t** does not stop its net)
- for each strongly connected component of the subgraph:
    - consider the maximal subset **S‘** $\subseteq$ **S** such that the subgraph generated by **S‘** is within the strongly connected component
    - If **S‘** is not empty then call this algorithm for **S‘**

Initially, call the algorithm for the set of all base solutions.

If it comes to ist proper end, then stop (**t** stops its net)

Strongly connected $\Rightarrow$ t does not stop its net

Can a Single Transition Stop an Entire Petri Net?

Jörg Desel
FernUniversität in Hagen
Germany