Verification of infinite state systems

Angelo Montanari and Gabriele Puppis

Department of Mathematics and Computer Science
University of Udine, Italy
{montana,puppis}@dimi.uniud.it

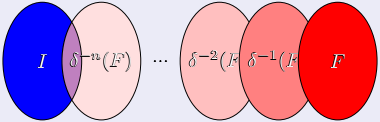
Go

In this part

We present solutions to the *plain reachability problem* for

- pushdown transition graphs
- transition graphs of Petri nets

Both solutions are based on **backward reachability analysis**.



Definition

Given an input-free pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$ and a set $X \subseteq Q \times \Gamma^*$ of configurations, we define

$$\delta^{-1}(X) := \{(q, w) \in Q \times \Gamma^* \ : \ \exists\, (q', w') \in X.\ (q, w) \xrightarrow[\mathcal{P}]{} (q', w')\}$$

Definition

Given an input-free pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$
and a set $X \subseteq Q \times \Gamma^*$ of configurations, we define

$$\delta^{-1}(X) := \{(q, w) \in Q \times \Gamma^* \ : \ \exists\, (q', w') \in X.\ (q, w) \xrightarrow[\mathcal{P}]{} (q', w')\}$$

Problem statement

Given an input-free pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$ and a set F of target configurations, compute the set $(\delta^{-1})^*(F)$ of all configurations $(q, w) \in Q \times \Gamma^*$ from which F is reachable.

In fact, we already know that the (full) model checking problem for pushdown transition systems is decidable. Here we give an efficient procedure to solve the reachability (sub-)problem.

The set $(\delta^{-1})^*(F)$ is the limit of the sequence

$$X_0 := F$$
$$X_{n+1} := X_n \cup \delta^{-1}(X_n)$$

Obviously, if $X_{n+1} = X_n$ for some $n \geq 0$, then $(\delta^{-1})^*(F) = X_n$.

The set $(\delta^{-1})^*(F)$ is the limit of the sequence

$$X_0 := F$$
$$X_{n+1} := X_n \cup \delta^{-1}(X_n)$$

Obviously, if $X_{n+1} = X_n$ for some $n \geq 0$, then $(\delta^{-1})^*(F) = X_n$.

Effectiveness

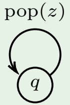
Two problems arise:

- the sets X_n may be infinite
  $\Rightarrow$ *symbolic representations* are needed
- $X_0, X_1, X_2, \ldots$ may be a strictly increasing sequence
  $\Rightarrow$ how to guarantee convergence in *finitely many* steps?

Example

Consider the pushdown system having

- a single state q
- a single stack symbol z
- a single transition (q, z, q, ε).



If we take $F = \{(q, \varepsilon)\}$, then

$$
\begin{aligned}
X_0 &= \{(q, \varepsilon)\} \\
X_1 &= \{(q, \varepsilon), (q, z)\} \\
X_2 &= \{(q, \varepsilon), (q, z), (q, zz)\} \\
&\dots
\end{aligned}
$$

We identify $\mathcal{P}$-configurations with finite words in $Q \cdot \Gamma^*$.

Moreover, to finitely represent sets of configurations of $\mathcal{P}$,
we restrict to regular sets of configurations
($\Rightarrow$ we represent them by finite state automata).

We identify $\mathcal{P}$-configurations with finite words in $Q \cdot \Gamma^*$.

Moreover, to finitely represent sets of configurations of $\mathcal{P}$,
we restrict to regular sets of configurations
($\Rightarrow$ we represent them by finite state automata).

To effectively solve the plain reachability problem, we compute $(\delta^{-1})^*(F)$ as the limit of another sequence $Y_0, Y_1, \ldots$ such that:

- (termination) $\exists\ n \geq 0.\ \ Y_{n+1} = Y_n$
- (completeness) $\forall\ n \geq 0.\ \ X_n \subseteq Y_n$
- (soundness) $\forall\ n \geq 0.\ \ Y_n \subseteq \bigcup_{i \geq 0} X_i$

We identify $\mathcal{P}$-configurations with finite words in $Q \cdot \Gamma^*$.

Moreover, to finitely represent sets of configurations of $\mathcal{P}$,
we restrict to regular sets of configurations
($\Rightarrow$ we represent them by finite state automata).

To effectively solve the plain reachability problem, we compute $(\delta^{-1})^*(F)$ as the limit of another sequence $Y_0, Y_1, \ldots$ such that:

- (termination) $\exists\ n \geq 0.\ \ Y_{n+1} = Y_n$
- (completeness) $\forall\ n \geq 0.\ \ X_n \subseteq Y_n$
- (soundness) $\forall\ n \geq 0.\ \ Y_n \subseteq \bigcup_{i \geq 0} X_i$

$\Rightarrow$ We shall define the sets Y_n as the languages recognized by suitable finite state automata $\mathcal{A}_n$ ...

Saturation algorithm (Bouajjani et al. '97)

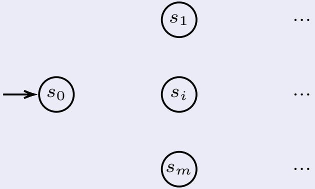
Sketch of the algorithm:

1. start with an automaton $\mathcal{A}_0$ with input alphabet $Q \cup \Gamma$ recognizing the regular language $Y_0 := X_0$ $(= F)$
2. build an automaton $\mathcal{A}_{n+1}$ recognizing Y_{n+1} by simply adding new transitions to $\mathcal{A}_n$
3. halt when $\mathcal{A}_{n+1} = \mathcal{A}_n$ (note: this eventually happens since only finitely many transitions can be added)

Assumptions on the initial automaton $\mathcal{A}_0$

W.l.o.g. we can assume that:

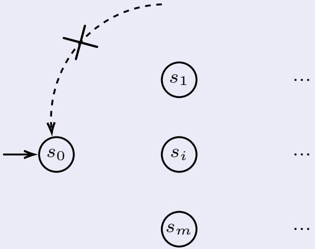
- the pushdown system $\mathcal{P}$ has m states $q_1, \ldots, q_m$
- the automaton $\mathcal{A}_0$ has a single initial non-final state s_0, m distinct states $s_1, \ldots, s_m$, and possibly other states



Assumptions on the initial automaton $\mathcal{A}_0$

W.l.o.g. we can assume that:

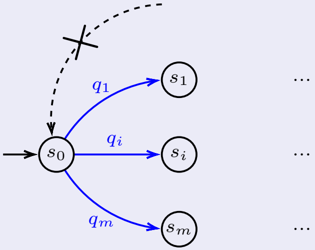
- the pushdown system $\mathcal{P}$ has m states $q_1, \ldots, q_m$
- the automaton $\mathcal{A}_0$ has a single initial non-final state s_0, m distinct states $s_1, \ldots, s_m$, and possibly other states
- there is no transition of $\mathcal{A}_0$ reaching the initial state s_0



Assumptions on the initial automaton $\mathcal{A}_0$

W.l.o.g. we can assume that:

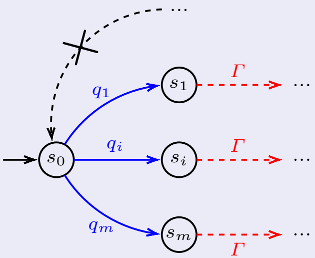
- the pushdown system $\mathcal{P}$ has m states $q_1, \ldots, q_m$
- the automaton $\mathcal{A}_0$ has a single initial non-final state s_0, m distinct states $s_1, \ldots, s_m$, and possibly other states
- there is no transition of $\mathcal{A}_0$ reaching the initial state s_0
- for $i = 1, \ldots, m$, the unique q_i-labeled transition is (s_0, q_i, s_i)



Assumptions on the initial automaton $\mathcal{A}_0$

W.l.o.g. we can assume that:

- the pushdown system $\mathcal{P}$ has m states $q_1, \ldots, q_m$
- the automaton $\mathcal{A}_0$ has a single initial non-final state s_0, m distinct states $s_1, \ldots, s_m$, and possibly other states
- there is no transition of $\mathcal{A}_0$ reaching the initial state s_0
- for $i = 1, \ldots, m$, the unique q_i-labeled transition is (s_0, q_i, s_i)
- other transitions are labeled by symbols in Γ



Definition (Construction of $\mathcal{A}_{n+1}$ from $\mathcal{A}_n$)

The automaton $\mathcal{A}_{n+1}$ for Y_{n+1} is obtained from $\mathcal{A}_n$
by adding, for each rule $(q_i, z, q_j, w') \in \Delta$,
a new transition (s_i, z, s') whenever $s_j \xrightarrow[\mathcal{A}_n]{w'} s'$.

Definition (Construction of $\mathcal{A}_{n+1}$ from $\mathcal{A}_n$)

The automaton $\mathcal{A}_{n+1}$ for Y_{n+1} is obtained from $\mathcal{A}_n$ by adding, for each rule $(q_i, z, q_j, w') \in \Delta$, a new transition (s_i, z, s') whenever $s_j \xrightarrow[\mathcal{A}_n]{w'} s'$.

Explanation

Assume that

- $\mathcal{A}_n$ reads $w' \in \Gamma^*$ from state s_j to state s'

Definition (Construction of $\mathcal{A}_{n+1}$ from $\mathcal{A}_n$)

The automaton $\mathcal{A}_{n+1}$ for Y_{n+1} is obtained from $\mathcal{A}_n$ by adding, for each rule $(q_i, z, q_j, w') \in \Delta$, a new transition (s_i, z, s') whenever $s_j \xrightarrow[\mathcal{A}_n]{w'} s'$.

Explanation

Assume that

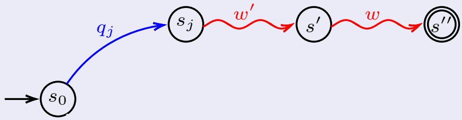
- $\mathcal{A}_n$ reads $w' \in \Gamma^*$ from state s_j to state s'
- $\mathcal{A}_n$ reads $w \in \Gamma^*$ from state s' to a final state s''

Definition (Construction of $\mathcal{A}_{n+1}$ from $\mathcal{A}_n$)

The automaton $\mathcal{A}_{n+1}$ for Y_{n+1} is obtained from $\mathcal{A}_n$
by adding, for each rule $(q_i, z, q_j, w') \in \Delta$,
a new transition (s_i, z, s') whenever $s_j \xrightarrow[\mathcal{A}_n]{w'} s'$.

Explanation

$\Rightarrow$ the $\mathcal{P}$-configuration $(q_j, w'w)$ belongs to Y_n.



Definition (Construction of $\mathcal{A}_{n+1}$ from $\mathcal{A}_n$)

The automaton $\mathcal{A}_{n+1}$ for Y_{n+1} is obtained from $\mathcal{A}_n$

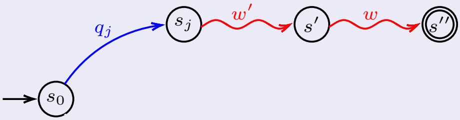
by adding, for each rule $(q_i, z, q_j, w') \in \Delta$,

a new transition (s_i, z, s') whenever $s_j \xrightarrow[\mathcal{A}_n]{w'} s'$.

Explanation

$\Rightarrow$ the $\mathcal{P}$-configuration $(q_j, w'w)$ belongs to Y_n.

Now, if $(q_i, z, q_j, w') \in \Delta$, then $(q_i, zw) \in Y_{n+1}$.



Definition (Construction of $\mathcal{A}_{n+1}$ from $\mathcal{A}_n$)

The automaton $\mathcal{A}_{n+1}$ for Y_{n+1} is obtained from $\mathcal{A}_n$

by adding, for each rule $(q_i, z, q_j, w') \in \Delta$,

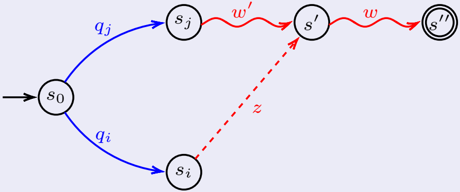
a new transition (s_i, z, s') whenever $s_j \xrightarrow[\mathcal{A}_n]{w'} s'$.

Explanation

$\Rightarrow$ the $\mathcal{P}$-configuration $(q_j, w'w)$ belongs to Y_n.

Now, if $(q_i, z, q_j, w') \in \Delta$, then $(q_i, zw) \in Y_{n+1}$.

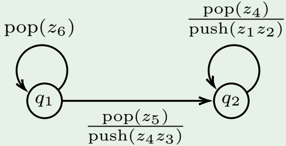
$\Rightarrow$ we can accept the $\mathcal{P}$-configuration (q_i, zw) by adding a z-labeled transition from s_i to s'.



Example

Consider the pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$, where

- $Q = \{q_1, q_2\}$
- $\Gamma = \{z_1, \ldots, z_6\}$
- $\Delta = \{(q_1, z_6, q_1, \varepsilon), (q_1, z_5, q_2, z_4 z_3), (q_2, z_4, q_2, z_1 z_2)\}$

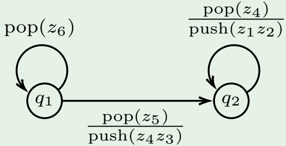


and the target set $F = \{(q_2, z_1 z_2 z_3)\}$.

Example

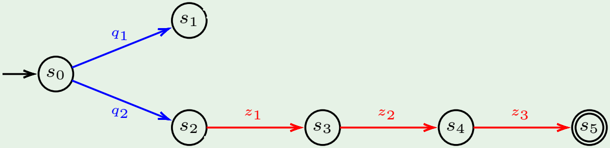
Consider the pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$, where

- $Q = \{q_1, q_2\}$
- $\Gamma = \{z_1, \ldots, z_6\}$
- $\Delta = \{(q_1, z_6, q_1, \varepsilon), (q_1, z_5, q_2, z_4 z_3),$
  $(q_2, z_4, q_2, z_1 z_2)\}$



and the target set $F = \{(q_2, z_1 z_2 z_3)\}$.

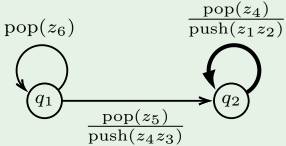
We represent $Y_0 = \{(q_2, z_1 z_2 z_3)\}$ with the automaton $\mathcal{A}_0$:



Example

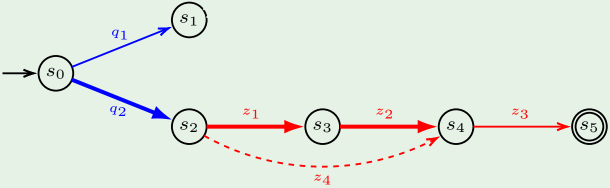
Consider the pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$, where

- $Q = \{q_1, q_2\}$
- $\Gamma = \{z_1, \ldots, z_6\}$
- $\Delta = \{(q_1, z_6, q_1, \varepsilon), (q_1, z_5, q_2, z_4 z_3),$
  $(q_2, z_4, q_2, z_1 z_2)\}$



and the target set $F = \{(q_2, z_1 z_2 z_3)\}$.

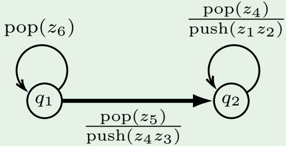
Since $s_0 \xrightarrow[\mathcal{A}_0]{q_2} s_2 \xrightarrow[\mathcal{A}_0]{z_1 z_2} s_4$ and $(q_2, z_4 w) \xrightarrow[\mathcal{P}]{} (q_2, z_1 z_2 w)$, we add a z_4-labeled transition from s_2 to s_4.



Example

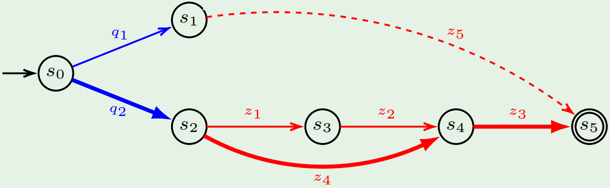
Consider the pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$, where

- $Q = \{q_1, q_2\}$
- $\Gamma = \{z_1, \ldots, z_6\}$
- $\Delta = \{(q_1, z_6, q_1, \varepsilon), (q_1, z_5, q_2, z_4 z_3), (q_2, z_4, q_2, z_1 z_2)\}$



and the target set $F = \{(q_2, z_1 z_2 z_3)\}$.

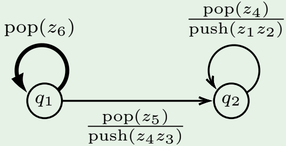
Since $s_0 \xrightarrow[\mathcal{A}_1]{q_2} s_2 \xrightarrow[\mathcal{A}_1]{z_4 z_3} s_4$ and $(q_1, z_5 w) \xrightarrow[\mathcal{P}]{} (q_2, z_4 z_3 w)$, we add a z_5-labeled transition from s_1 to s_4.



Example

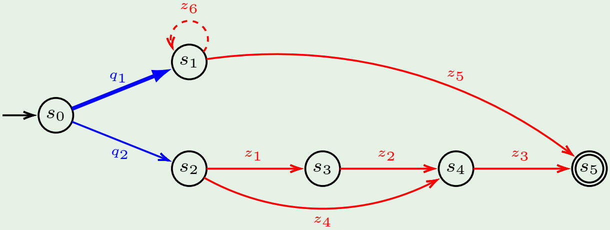
Consider the pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$, where

- $Q = \{q_1, q_2\}$
- $\Gamma = \{z_1, \ldots, z_6\}$
- $\Delta = \{(q_1, z_6, q_1, \varepsilon), (q_1, z_5, q_2, z_4 z_3),$
  $(q_2, z_4, q_2, z_1 z_2)\}$



and the target set $F = \{(q_2, z_1 z_2 z_3)\}$.

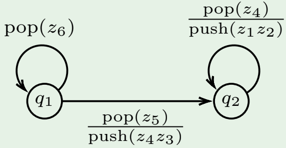
Since $s_0 \xrightarrow[\mathcal{A}_2]{q_1} s_1 \xrightarrow[\mathcal{A}_2]{\varepsilon} s_1$ and $(q_1, z_6 w) \xrightarrow[\mathcal{P}]{} (q_1, w)$,
we add a z_6-labeled transition from s_1 to s_1.



Example

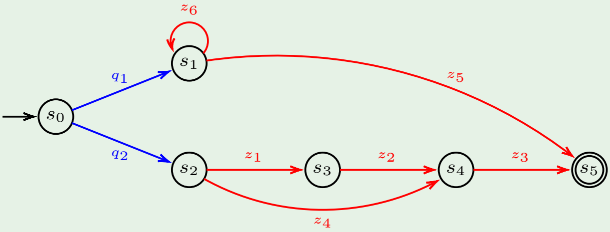
Consider the pushdown system $\mathcal{P} = (Q, \Gamma, \Delta)$, where

- $Q = \{q_1, q_2\}$
- $\Gamma = \{z_1, \ldots, z_6\}$
- $\Delta = \{(q_1, z_6, q_1, \varepsilon), (q_1, z_5, q_2, z_4 z_3), (q_2, z_4, q_2, z_1 z_2)\}$



and the target set $F = \{(q_2, z_1 z_2 z_3)\}$.

No more transitions can be added.
Thus $(\delta^{-1})^*(F) = \{(q_2, z_1 z_2 z_3), (q_2, z_4 z_3), (q_1, z_6^* z_5)\}$.



Note:

- the conditions for adding new transitions can be effectively tested in polynomial time
- at most a polynomial number of transitions are added to the initial automaton $\mathcal{A}_0$

This gives an efficient (polynomial-time) algorithm that solves the **(existential) plain reachability problem** for pushdown graphs

(EFψ '*there exist a path along which* ψ *eventually holds*')

Variants of the plain reachability problem can be considered.

For instance, **universal reachability problem**:
AFψ '*every infinite path eventually satisfies* ψ'

Variants of the plain reachability problem can be considered.

For instance, **universal reachability problem**:
AFψ '*every infinite path eventually satisfies* ψ'

Alternating reachability problem

The **alternating reachability problem** is a generalization of reachability problems, where existential *EF* and universal *AF* quantifications can be paired.

It can be viewed as a game played over a graph by two players *A* and *B*:

- *A* wants to reach a safe region *F*
- *B* wants to indefinitely delay this achievement.

Instances of the alternating reachability problem over pushdown transition graphs are naturally encoded by

alternating pushdown systems

which are able to spawn different computations at the same time (*existential non-determinism* and *universal non-determinism*).

Instances of the alternating reachability problem over pushdown transition graphs are naturally encoded by

alternating pushdown systems

which are able to spawn different computations at the same time (*existential non-determinism* and *universal non-determinism*).

Generalization of the saturation algorithm

A generalization of the saturation algorithm for the alternating reachability problem over pushdown systems can be given.

Such a generalization uses **alternating finite state automata**, rather than classical (non-deterministic) finite state automata, to represent increasing sets $Y_0, Y_1, Y_2, \ldots$ of configurations.

Other generalizations of the saturation algorithm have been studied for

- **higher-order pushdown systems** [Bouajjani and Meyer '04] (i.e., pushdown systems working on level n stacks)
- **ground tree rewriting systems** [Löding '06] (i.e., rewriting systems working on finite colored trees)
- **bifix rewriting systems** [Altenbernd and Thomas] (they are similar to pushdown automata, but rewriting may occur at the top or at the bottom of the stack in a non-deterministic way)

We consider now Petri nets.

Definition (Petri net)

A **Petri net** is a tuple $\mathcal{P} = (P, T, I, O)$, where

- P is a finite set of **places**
- T is a finite set of **transitions**
- $I : P \times T \rightarrow \mathbb{N}$ is the **input arc function** specifying how many arcs go from $p \in P$ to $t \in T$
- $O : T \times P \rightarrow \mathbb{N}$ is the **output arc function** specifying how many arcs go from $t \in T$ to $p \in P$

Intuitively: the above definition is nothing but a specification of a

non-simple directed bipartite graph.

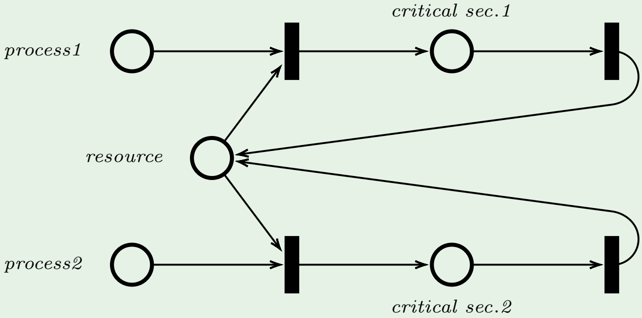
How do they work?

Basic ingredients:

- a **configuration** is a function $m : P \rightarrow \mathbb{N}$
  (it assigns a certain number of tokens to each place)
- a transition t is **enabled** in a configuration m
  if $m(p) \geq I(p, t)$ for every place p
  (namely, if each place p contains at least $I(p, t)$ tokens)
- when a transition t fires, the **next configuration** m' is
  such that $m'(p) = m(p) - I(p, t) + O(t, p)$ for all $p \in P$
  (namely, $I(p, t)$ tokens are consumed from p and, at the same time, $O(t, p)$ tokens are produced in p)

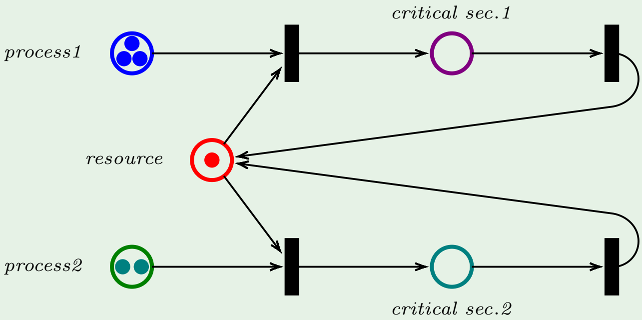
Example

The following Petri net models two parallel processes competing for a shared resource:



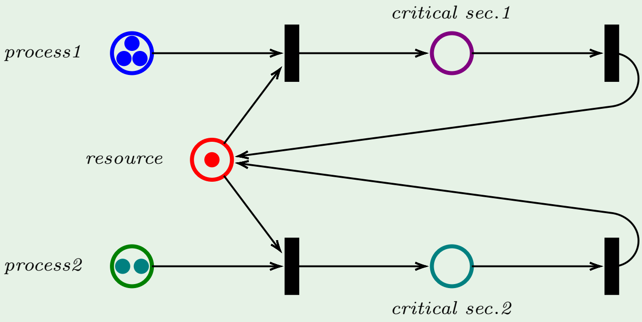
Example

The initial configuration m_0 is encoded by the tuple $[3, 1, 2, 0, 0]$



Example

When a transition fires, the tokens in the input places are consumed and new ones are produced inside output places.

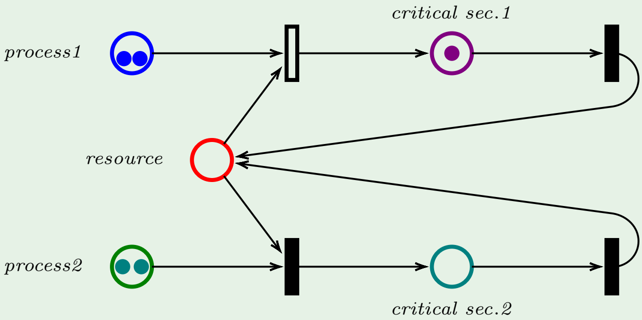


Example of computation:

$[3, 1, 2, 0, 0]$

Example

When a transition fires, the tokens in the input places are consumed and new ones are produced inside output places.

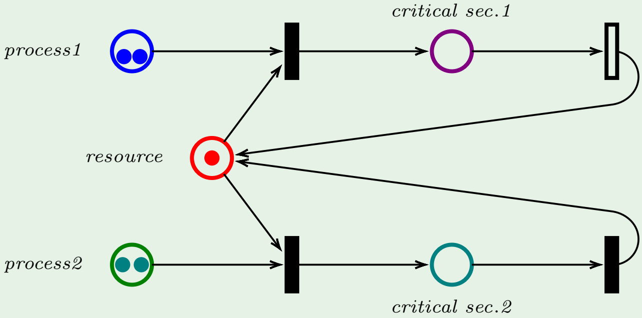


Example of computation:

$[3, 1, 2, 0, 0] \xrightarrow[\mathcal{P}]{} [2, 0, 2, 1, 0]$

Example

When a transition fires, the tokens in the input places are consumed and new ones are produced inside output places.

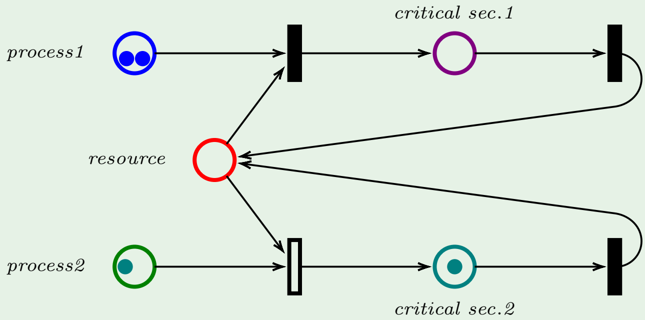


Example of computation:

$[3, 1, 2, 0, 0] \xrightarrow[\mathcal{P}]{} [2, 0, 2, 1, 0] \xrightarrow[\mathcal{P}]{} [2, 1, 2, 0, 0]$

Example

When a transition fires, the tokens in the input places are consumed and new ones are produced inside output places.

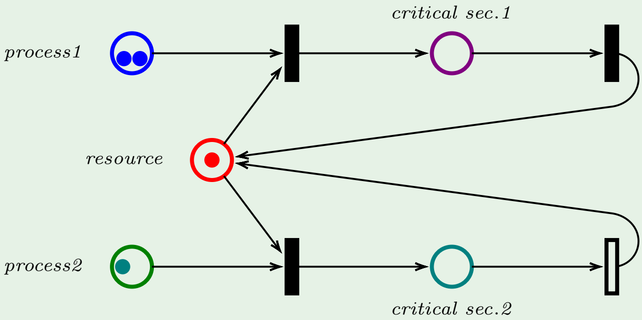


Example of computation:

$$[3, 1, 2, 0, 0] \xrightarrow[\mathcal{P}]{} [2, 0, 2, 1, 0] \xrightarrow[\mathcal{P}]{} [2, 1, 2, 0, 0] \xrightarrow[\mathcal{P}]{} [2, 0, 1, 0, 1]$$

Example

When a transition fires, the tokens in the input places are consumed and new ones are produced inside output places.



Example of computation:

$$[3,1,2,0,0] \xrightarrow[\mathcal{P}]{} [2,0,2,1,0] \xrightarrow[\mathcal{P}]{} [2,1,2,0,0] \xrightarrow[\mathcal{P}]{} [2,0,1,0,1] \xrightarrow[\mathcal{P}]{} [2,1,1,0,0]$$

Definition (Petri net transition graph)

The **transition graph** of a Petri net $\mathcal{P} = (P, T, I, O)$ is the transition system $\mathcal{T} = (\mathbb{N}^P, \delta)$, where

- $\mathbb{N}^P$ is the set of all possible configurations $m : P \to \mathbb{N}$
- δ is the transition relation such that $(m, m') \in \delta$ iff $\exists\, t \in T.\ \forall\, p \in P.\quad m'(p) = m(p) - I(p,t) + O(t,p)$.

Definition (Petri net transition graph)

The **transition graph** of a Petri net $\mathcal{P} = (P, T, I, O)$ is the transition system $\mathcal{T} = (\mathbb{N}^P, \delta)$, where

- $\mathbb{N}^P$ is the set of all possible configurations $m : P \to \mathbb{N}$
- δ is the transition relation such that $(m, m') \in \delta$ iff $\exists\, t \in T.\ \forall\, p \in P.\quad m'(p) = m(p) - I(p,t) + O(t,p)$.

Problem statement

Given a Petri net $\mathcal{P} = (P, T, I, O)$ and a set F of target configurations, we want to compute the set $(\delta^{-1})^*(F)$ of *all configurations $m : P \to \mathbb{N}$ from which F is reachable.*

Like in the case of pushdown transition graphs, we cannot finitely represent all possible sets of configurations of a Petri net $\mathcal{P}$ (note: there are uncountably many of them).

$\Rightarrow$ In order to find effective solutions to the reachability problem, we must restrict to a proper subclass of sets of $\mathcal{P}$-configurations.

Before introducing (finitely representable) sets of configurations, we give a notion of **partial order** over the configurations of $\mathcal{P}$.

Definition (Partial order on Petri net configurations)

Given a Petri net $\mathcal{P} = (P, T, I, O)$, we define the **partial order** $\leq$ on $\mathcal{P}$-configurations such that

$$m \leq m' \quad \text{iff, for all places } p, \quad m(p) \leq m'(p)$$

Definition (Partial order on Petri net configurations)

Given a Petri net $\mathcal{P} = (P, T, I, O)$, we define the **partial order** $\leq$ on $\mathcal{P}$-configurations such that

$$m \leq m' \quad \text{iff, for all places } p, \quad m(p) \leq m'(p)$$

Example

$[2, 1, 2, 0, 0] \leq [3, 1, 5, 0, 0]$ and $[1, 0, 2, 1, 0] \not\leq [5, 2, 2, 0, 0]$

Definition (Partial order on Petri net configurations)

Given a Petri net $\mathcal{P} = (P, T, I, O)$, we define the **partial order** $\leq$ on $\mathcal{P}$-configurations such that

$$m \leq m' \quad \text{iff, for all places } p, \quad m(p) \leq m'(p)$$

Example

$[2, 1, 2, 0, 0] \leq [3, 1, 5, 0, 0]$ and $[1, 0, 2, 1, 0] \not\leq [5, 2, 2, 0, 0]$

Basic property

The partial order $\leq$ is actually a **well** partial order:

- there are no infinite sequences of strictly decreasing elements $m_1 > m_2 > m_3 > \ldots$
- there are no infinite sequences of pairwise incomparable elements $\forall\, i \neq j.\ m_i \not\leq m_j$

Definition (Upward closed set)

Hereafter, we restrict to **upward closed sets of configurations**, namely, sets $X \subseteq \mathbb{N}^P$ such that, for all $m, m' : P \to \mathbb{N}$,

$$m \in X \ \wedge \ m \leq m' \ \rightarrow \ m' \in X$$

Definition (Upward closed set)

Hereafter, we restrict to **upward closed sets of configurations**, namely, sets $X \subseteq \mathbb{N}^P$ such that, for all $m, m' : P \rightarrow \mathbb{N}$,

$$m \in X \ \wedge \ m \leq m' \ \rightarrow \ m' \in X$$

Definition (Minor set)

The **minor set** $min(X)$ of an upward closed set X is

$$min(X) := \{m \in X \ : \ \nexists \ m' \in X.\ m' \leq m\}$$

(intuitively, $min(X)$ consists of all the minimal elements of X)

Note: the minor set $min(X)$ uniquely determines X, since $X = min(X)\!\uparrow$, where $Y\!\uparrow := \{m \in \mathbb{N}^P \ : \ \exists \ m' \in Y.\ m' \leq m\}$

Why upward closed sets and minor sets?

Noticeable properties:

1. minor sets are finite objects
   $\Rightarrow$ they are representations of upward closed sets
2. Petri nets are monotone systems w.r.t. $\leq$
3. upward closed sets are closed under δ^{-1}
4. there are no infinite sequences of strictly increasing upward closed sets $X_0 \subset X_1 \subset X_2 \subset \ldots$.

Property 1

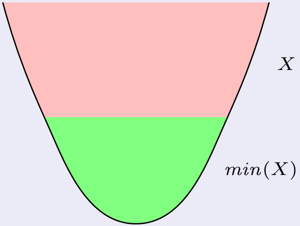
Minor sets are finite ($\Rightarrow$ representations of upward closed sets).

Property 1

Minor sets are finite ($\Rightarrow$ representations of upward closed sets).

Proof

Consider an upward closed set X and its minor set $min(X)$.

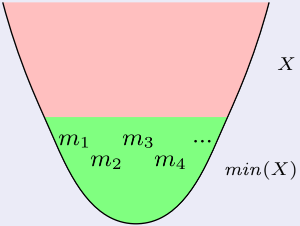


Property 1

Minor sets are finite ($\Rightarrow$ representations of upward closed sets).

Proof

min(X) consists of pairwise incomparable elements $m_1, m_2, ...$

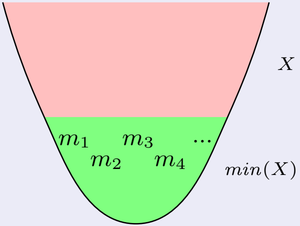


Property 1

Minor sets are finite ($\Rightarrow$ representations of upward closed sets).

Proof

Since $\leq$ is a well partial order, *min*(X) is finite.



Property 2

Petri nets are **monotone systems** w.r.t. $\leq$, namely

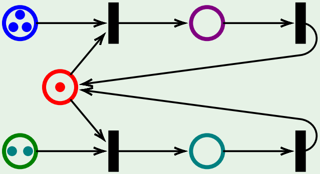
$$\begin{cases} m_1 \xrightarrow[\mathcal{P}]{} m_2 \\ m_1 \leq m_1' \end{cases} \Rightarrow \quad \exists\, m_2' : P \rightarrow \mathbb{N}. \begin{cases} m_1' \xrightarrow[\mathcal{P}]{} m_2' \\ m_2 \leq m_2' \end{cases}$$

Property 2

Petri nets are **monotone systems** w.r.t. $\leq$, namely

$$\begin{cases} m_1 \xrightarrow[\mathcal{P}]{} m_2 \\ m_1 \leq m'_1 \end{cases} \Rightarrow \quad \exists\, m'_2 : P \to \mathbb{N}.\ \begin{cases} m'_1 \xrightarrow[\mathcal{P}]{} m'_2 \\ m_2 \leq m'_2 \end{cases}$$

Example



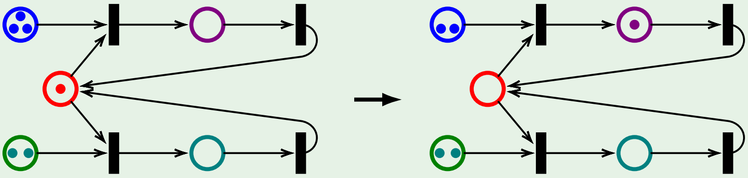
$[3, 1, 2, 0, 0]$

Property 2

Petri nets are **monotone systems** w.r.t. $\leq$, namely

$$\begin{cases} m_1 \xrightarrow[\mathcal{P}]{} m_2 \\ m_1 \leq m_1' \end{cases} \Rightarrow \quad \exists\, m_2' : P \to \mathbb{N}. \begin{cases} m_1' \xrightarrow[\mathcal{P}]{} m_2' \\ m_2 \leq m_2' \end{cases}$$

Example



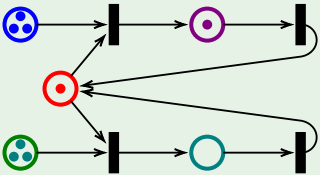
$[3, 1, 2, 0, 0] \longrightarrow [2, 0, 2, 1, 0]$

Property 2

Petri nets are **monotone systems** w.r.t. $\leq$, namely

$$\begin{cases} m_1 \xrightarrow[\mathcal{P}]{} m_2 \\ m_1 \leq m_1' \end{cases} \Rightarrow \quad \exists\, m_2' : P \to \mathbb{N}. \begin{cases} m_1' \xrightarrow[\mathcal{P}]{} m_2' \\ m_2 \leq m_2' \end{cases}$$

Example



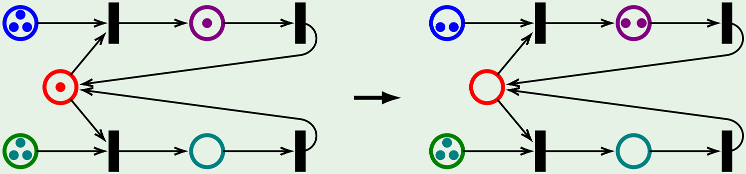
$$\begin{array}{ccc} [3, 1, 2, 0, 0] & \longrightarrow & [2, 0, 2, 1, 0] \\ \leq & & \\ [3, 1, 3, 1, 0] & & \end{array}$$

Property 2

Petri nets are **monotone systems** w.r.t. $\leq$, namely

$$\begin{cases} m_1 \xrightarrow[\mathcal{P}]{} m_2 \\ m_1 \leq m_1' \end{cases} \Rightarrow \quad \exists\, m_2' : P \to \mathbb{N}. \begin{cases} m_1' \xrightarrow[\mathcal{P}]{} m_2' \\ m_2 \leq m_2' \end{cases}$$

Example



$$\begin{array}{ccc} [3,1,2,0,0] & \longrightarrow & [2,0,2,1,0] \\ \leq & & \leq \\ [3,1,2,0,0] & \longrightarrow & [2,0,3,2,0] \end{array}$$

Property 3

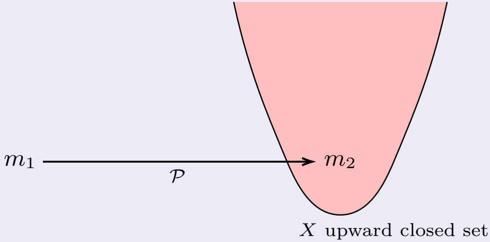
Upward closed sets are closed under δ^{-1}.

Property 3

Upward closed sets are closed under δ^{-1}.

Proof

Consider an upward closed set X and
a configuration $m_2 \in X$ with its pre-image m_1.

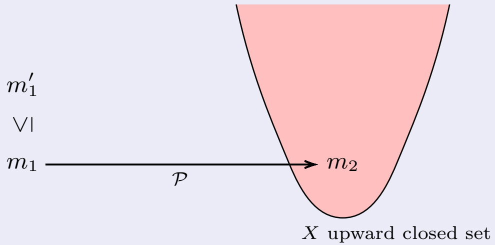


Property 3

Upward closed sets are closed under δ^{-1}.

Proof

Let m_1' be another configuration above m_1.

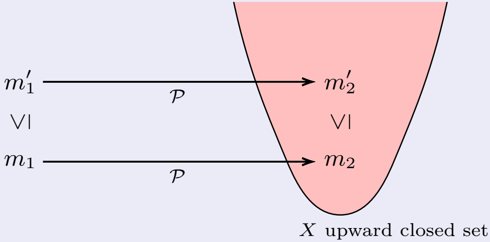


Property 3

Upward closed sets are closed under δ^{-1}.

Proof

From monotonicity of Petri nets, there is a configuration m_2' above m_2 which is the image of m_1' under δ.

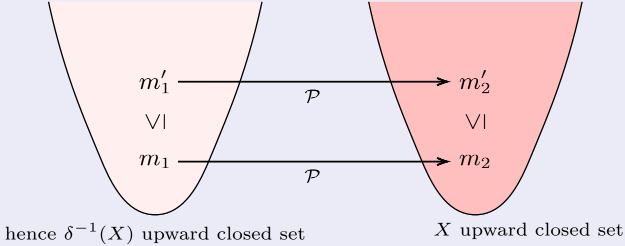


Property 3

Upward closed sets are closed under δ^{-1}.

Proof

This implies that $\delta^{-1}(X)$ is an upward closed set.



Property 4

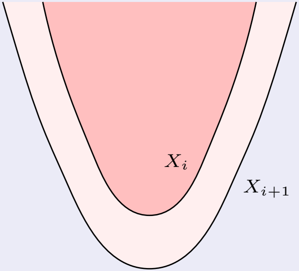
There are no infinite sequences of strictly increasing upward closed sets $X_0 \subset X_1 \subset X_2 \subset \ldots$.

Property 4

There are no infinite sequences of strictly increasing upward closed sets $X_0 \subset X_1 \subset X_2 \subset \ldots$.

Proof sketch

Consider two sets X_i and X_{i+1}, with $X_i \subset X_{i+1}$.

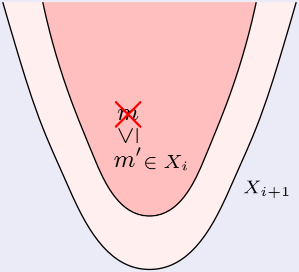


Property 4

There are no infinite sequences of strictly increasing upward closed sets $X_0 \subset X_1 \subset X_2 \subset \ldots$.

Proof sketch

Take a configuration m such that $m \in X_{i+1} \setminus X_i$.



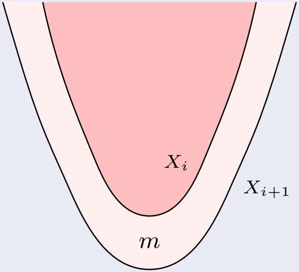
Property 4

There are no infinite sequences of strictly increasing upward closed sets $X_0 \subset X_1 \subset X_2 \subset \ldots$.

Proof sketch

Take a configuration m such that $m \in X_{i+1} \setminus X_i$.

$\Rightarrow$ either m is below some element of X_i



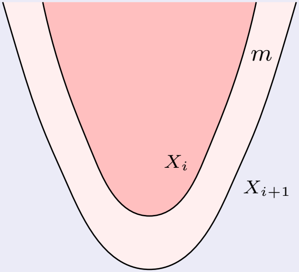
Property 4

There are no infinite sequences of strictly increasing upward closed sets $X_0 \subset X_1 \subset X_2 \subset \ldots$.

Proof sketch

Take a configuration m such that $m \in X_{i+1} \setminus X_i$.

$\Rightarrow$ either m is below some element of X_i
or m is incomparable with any element of X_i



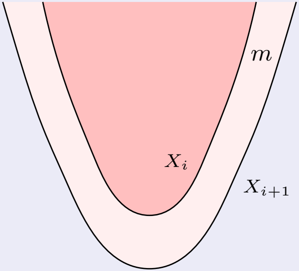
Property 4

There are no infinite sequences of strictly increasing upward closed sets $X_0 \subset X_1 \subset X_2 \subset \dots$.

Proof sketch

Since $\leq$ is a *well* partial order, neither infinite decreasing chains nor infinite antichains are allowed,

$\Rightarrow$ none of the above two cases may occur infinitely often.



Theorem (Parosh et al. '00)

Given the transition graph $\mathcal{T} = (\mathbb{N}^P, \delta)$ of a Petri net and an upward closed set F of configurations, the minor set of $(\delta^{-1})^*(F)$ can be effectively calculated as follows:

1. *start with $Y_0 := min(F)$*
2. *compute $Y_{i+1} := min\big(Y_i{\uparrow} \,\cup\, \delta^{-1}(Y_i{\uparrow})\big)$*
3. *if $Y_{i+1} = Y_i$, then $(\delta^{-1})^*(F) = Y_i{\uparrow}$.*

Theorem (Parosh et al. '00)

Given the transition graph $\mathcal{T} = (\mathbb{N}^P, \delta)$ of a Petri net and an upward closed set F of configurations, the minor set of $(\delta^{-1})^*(F)$ can be effectively calculated as follows:

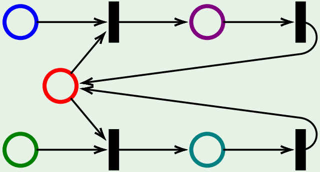
1. *start with $Y_0 := min(F)$*
2. *compute $Y_{i+1} := min\big(Y_i\uparrow \ \cup \ \delta^{-1}(Y_i\uparrow)\big)$*
3. *if $Y_{i+1} = Y_i$, then $(\delta^{-1})^*(F) = Y_i\uparrow$.*

Corollary

The plain reachability problem over transition graphs of Petri nets restricted to upward closed sets is decidable.

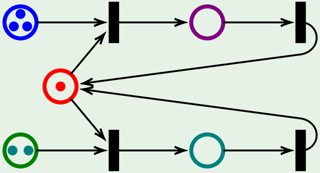
Example

We want to check the following **mutual exclusion property**:
'if the red place is initialized with one token and the violet and the cyan places with zero tokens, then it will never happen that both the violet and the cyan places have tokens at the same time'.



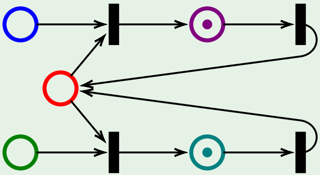
Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \, : \, x, y \geq 0\}$



Example

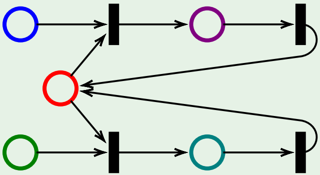
- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \;:\; x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \;:\; x, y, z \geq 0 \;\wedge\; u, v \geq 1\}$
  (note: F is infinite but **upward closed**)



Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \,:\, x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \,:\, x, y, z \geq 0 \,\wedge\, u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$

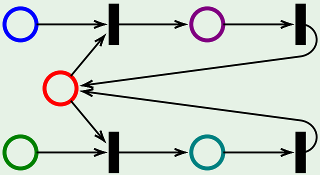


$Y_0 = \{[0, 0, 0, 1, 1]\}$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \; : \; x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \; : \; x, y, z \geq 0 \; \wedge \; u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$



$$Y_1 = \left\{ \begin{array}{c} [0, 0, 0, 1, 1] \\ \\ \end{array} \right\}$$

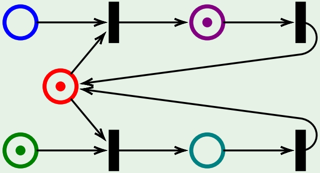
$$\downarrow$$

$$Y_0 = \{[0, 0, 0, 1, 1]\}$$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \,:\, x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \,:\, x, y, z \geq 0 \;\wedge\; u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$



$$Y_1 = \left\{ \begin{array}{c} [0, 0, 0, 1, 1] \\ \\ \end{array} \right\}$$

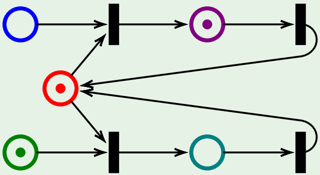
$$\downarrow$$

$$Y_0 = \{[0, 0, 0, 1, 1]\}$$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \;:\; x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \;:\; x, y, z \geq 0 \;\wedge\; u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$

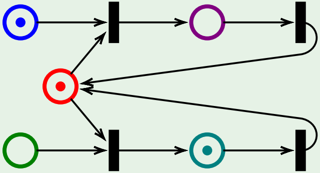


$$Y_1 = \left\{ \begin{array}{l} [0, 0, 0, 1, 1] \\ [0, 1, 1, 1, 0] \\ \end{array} \right\}$$
$$\downarrow$$
$$Y_0 = \{[0, 0, 0, 1, 1]\}$$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \,:\, x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \,:\, x, y, z \geq 0 \;\wedge\; u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$



$$Y_1 = \left\{ \begin{matrix} [0, 0, 0, 1, 1] \\ [0, 1, 1, 1, 0] \\ \end{matrix} \right\}$$

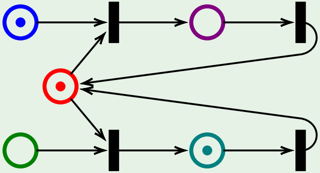
$$\downarrow$$

$$Y_0 = \{[0, 0, 0, 1, 1]\}$$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \ : \ x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \ : \ x, y, z \geq 0 \ \wedge \ u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$

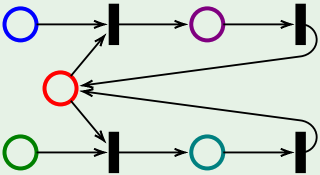


$$Y_1 = \left\{ \begin{array}{c} [0, 0, 0, 1, 1] \\ [0, 1, 1, 1, 0] \\ [1, 1, 0, 0, 1] \end{array} \right\}$$
$$\downarrow$$
$$Y_0 = \{[0, 0, 0, 1, 1]\}$$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \,:\, x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \,:\, x, y, z \geq 0 \,\wedge\, u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$



$$Y_2 = \left\{\begin{array}{l}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\\ \\ \end{array}\right\}$$
$$\downarrow$$
$$Y_1 = \left\{\begin{array}{l}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\end{array}\right\}$$
$$\downarrow$$
$$Y_0 = \{[0,0,0,1,1]\}$$

Example

- Set of initial configurations:

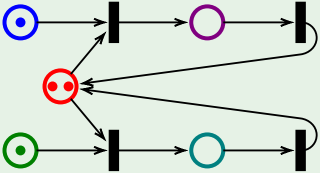
  $I = \{[x, 1, y, 0, 0] \ : \ x, y \geq 0\}$

- Set of bad configurations:

  $F = \{[x, y, z, u, v] \ : \ x, y, z \geq 0 \ \wedge \ u, v \geq 1\}$

  (note: F is infinite but **upward closed**)

We use backward reachability analysis to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$



$$Y_2 = \left\{\begin{array}{c}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\\ \\ \end{array}\right\}$$

$$\downarrow$$

$$Y_1 = \left\{\begin{array}{c}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\end{array}\right\}$$

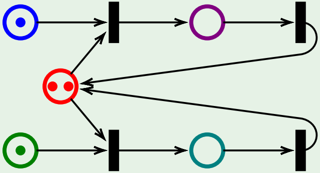
$$\downarrow$$

$$Y_0 = \{[0,0,0,1,1]\}$$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \ : \ x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \ : \ x, y, z \geq 0 \ \wedge \ u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis
to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$



$$Y_2 = \left\{\begin{matrix}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\\ [1,2,1,0,0]\end{matrix}\right\}$$

$$\downarrow$$

$$Y_1 = \left\{\begin{matrix}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\end{matrix}\right\}$$

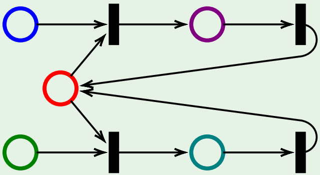
$$\downarrow$$

$$Y_0 = \{[0,0,0,1,1]\}$$

Example

- Set of initial configurations:
  $I = \{[x, 1, y, 0, 0] \;:\; x, y \geq 0\}$
- Set of bad configurations:
  $F = \{[x, y, z, u, v] \;:\; x, y, z \geq 0 \;\wedge\; u, v \geq 1\}$
  (note: F is infinite but **upward closed**)

We use backward reachability analysis to check whether $(\delta^{-1})^*(F) \cap I \neq \emptyset$



$$Y_3 = \left\{\begin{array}{l}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\\ [1,2,1,0,0]\end{array}\right\}$$
$$\downarrow$$
$$Y_2 = \left\{\begin{array}{l}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\\ [1,2,1,0,0]\end{array}\right\}$$
$$\downarrow$$
$$Y_1 = \left\{\begin{array}{l}[0,0,0,1,1]\\ [0,1,1,1,0]\\ [1,1,0,0,1]\end{array}\right\}$$
$$\downarrow$$
$$Y_0 = \{[0,0,0,1,1]\}$$

What did we use?

- a well partial order $\leq$ on the configurations
- monotonicity of transition systems w.r.t. $\leq$
- computability of $min\big(\delta^{-1}(Y\uparrow)\big)$ for any minor set Y.

What did we use?

- a well partial order $\leq$ on the configurations
- monotonicity of transition systems w.r.t. $\leq$
- computability of $min\left(\delta^{-1}(Y\uparrow)\right)$ for any minor set Y.

The first requirement can be relaxed:
a **well quasi-order** is sufficient !

Definition (Quasi-order)

A **quasi-order** is a reflexive and transitive relation $\preceq$
(it may happen that $x \neq y \ \wedge \ x \preceq y \ \wedge \ y \preceq x$).

Moreover, $\preceq$ is a **well** quasi-order if for every infinite sequence $x_0, x_1, x_2, \ldots$, there are $i < j$ such that $x_i \preceq x_j$.

Definition (Well-structured transition system)

A transition system $\mathcal{T} = (S, \delta)$ is **well-structured** if

- there is a well quasi-order $\preceq$ on S
- δ is monotone w.r.t. $\preceq$
- $min\big(\delta^{-1}(Y\uparrow)\big)$ is computable for any minor set Y.

Definition (Well-structured transition system)

A transition system $\mathcal{T} = (S, \delta)$ is **well-structured** if

- there is a well quasi-order $\preceq$ on S
- δ is monotone w.r.t. $\preceq$
- $min\big(\delta^{-1}(Y\uparrow)\big)$ is computable for any minor set Y.

Theorem (Parosh et al. '00)

Backward reachability analysis of well-structured systems (starting from upward closed sets) is effective.

Examples of well-structured systems

- **timed Petri nets**:
  every token has an *age* and every arc has *constraints* on ages
- **lossy channel systems**:
  they model finite state processes communicating via
  unreliable (they can lose messages) FIFO channels
- **basic parallel processes**:
  these are rewriting systems where words are viewed as *multi-sets*
  and rewritings may involve symbols at *non-contiguous positions*
- **real time automata**:
  they are equipped with *counters over real values*, inequalities
  (e.g., $x < c$, $x \leq c$, $x > c$, $x \geq c$) associated with states and
  transitions, and reset operations ($x := 0$) on transitions
- **integral relational automata**:
  they are equipped with *counters over integers*, inequalities
  (e.g., $x < y$, $x < c$, $x > c$) associated with states and transitions,
  and update operations (e.g., $x := y$, $x := c$, $x := ?$).

Go