Algorithmic Program Verification

Parosh Aziz Abdulla
Uppsala University
Sweden
Algorithmic Program Verification

Model Checking of Infinite State-Systems

Parosh Aziz Abdulla
Uppsala University
Sweden
Model Checking
Model $\models$ (safety) property
Background

Classical Approach
Finite-State Systems

Model Checking
Model $\models$ (safety) property
Background

Classical Approach
Finite-State Systems

Model Checking
Model $\models$ (safety) property

Challenge:
Infinite-State Systems
Background

Classical Approach
Finite-State Systems

Model Checking
\[ \text{Model} \models \text{(safety) property} \]

Challenge:
Infinite-State Systems

Sources of “Infiniteness”: 
Background

**Classical Approach**
Finite-State Systems

**Challenge:**
Infinite-State Systems

**Model Checking**
Model $\models$ (safety) property

**Sources of “Infiniteness”:**

Unbounded Data Structures
- stacks (recursion)
- queues (protocols)
- counters (programs)
- clocks (time)
- lists, trees, graphs (heaps)
Background

Classical Approach
Finite-State Systems

Model Checking
Model \models \text{(safety) property}

Challenge:
Infinite-State Systems

Sources of “Infiniteness”:

Unbounded Data Structures
- stacks (recursion)
- queues (protocols)
- counters (programs)
- clocks (time)
- lists, trees, graphs (heaps)

Unbounded Control Structures
- parameterized systems
- multithreaded programs
- concurrent libraries
- Petri nets
Background

Classical Approach
Finite-State Systems

Multiple Sources:
• timed Petri nets
• recursive programs with unbounded data
• channels with time stamps
• etc

Model Checking
Model $\models$ (safety) property

Challenge:
Infinite-State Systems

Sources of “Infiniteness”:

Unbounded Data Structures
• stacks (recursion)
• queues (protocols)
• counters (programs)
• clocks (time)
• lists, trees, graphs (heaps)

Unbounded Control Structures
• parameterized systems
• multithreaded programs
• concurrent libraries
• Petri nets
Infinite-State Systems
Infinite-State Systems

Unbounded Number of Processes

Cache Coherence Protocol

P

cache

cache

cache

P

cache
Infinite-State Systems

Unbounded Number of Processes

Cache Coherence Protocol

Diagram:

- Four blue circles labeled P
- Four red boxes labeled cache
- Connections between P and cache boxes
- Wavy lines indicating coherence protocol

Infinite-State Systems

Unbounded Number of Processes

Cache Coherence Protocol
Cache Coherence Protocol

- unbounded number of processes
- correctness:
  - exclusive ownership: at most one process
Infinite-State Systems

Unbounded Data Structures
Infinite-State Systems

Unbounded Data Structures

Unbounded Channels
Infinite-State Systems

Unbounded Data Structures

Unbounded Channels
Infinite-State Systems

Unbounded Data Structures

Unbounded Channels
Infinite-State Systems

Unbounded Data Structures

Unbounded Channels
Infinite-State Systems

Unbounded Data Structures

Unbounded Channels

- unbounded FIFO channels
- correctness:
  - regardless of channels size
Infinite-State Systems

Unbounded Data Structures

Unbounded Stack

push (a);

push (b);

push (c);

pop ( );

push (a);

push (d);
Infinite-State Systems
Unbounded Data Structures

Unbounded Stack

- push (a);
- push (b);
- push (c);
- pop ();
- push (d);

- unbounded stack
- correctness:
  - regardless of stack size
Infinite-State Systems

Unbounded Data Structures

Unbounded Counters

\[ c \rightarrow c++ \]
\[ d \rightarrow d++ \]
\[ d \rightarrow d=-0? \]

Graphical representation of state transitions and operations.
Infinite-State Systems

Unbounded Data Structures

Unbounded Counters

- unbounded counters
- correctness:
  - regardless of counter values
Infinite-State Systems
Unbounded Data Structures
Clocks
Infinite-State Systems
Unbounded Data Structures

Clocks

- timed systems
- real-value clocks
Background

Classical Approach
Finite-State Systems

Model Checking
Model $\models$ (safety) property

Challenge:
Infinite-State Systems

Sources of "Infiniteness":

Unbounded Data Structures
- stacks (recursion)
- queues (protocols)
- counters (programs)
- clocks (time)
- lists, trees, graphs (heaps)

Unbounded Control Structures
- parameterized systems
- multithreaded programs
- concurrent libraries
- Petri nets

Multiple Sources:
- timed Petri nets
- recursive programs with unbounded data
- queues with time stamps
- etc
Background

Classical Approach
Finite-State Systems

Model Checking
Model $\models$ (safety) property

Challenge:
Infinite-State Systems

Sources of “Infiniteness”:

Unbounded Data Structures
- stacks (recursion)
- queues (protocols)
- counters (programs)
- clocks (time)
- lists, trees, graphs (heaps)

Unbounded Control Structures
- parameterized systems
- multithreaded programs
- concurrent libraries
- Petri nets

Multiple Sources:
- timed Petri nets
- recursive programs with unbounded data
- queues with time stamps
- etc
Parameterized Systems
Parameterized Systems
• Specification
  • Mutual Exclusion (MutEx):
  • At most one process in C
Parameterized Systems

\[ P^n | L \]

- Specification
  - Mutual Exclusion (MutEx):
  - At most one process in C
Parameterized Systems

Task = Parameterized Verification

Verify correctness regardless of the number of processes

\( \forall n. (P^n | L) \models \text{MutEx} \)

Specification

- Mutual Exclusion (MutEx):
  - At most one process in C

\( P^n | L \)
Parameterized Systems

- **Task = Parameterized Verification**
  - Verify correctness regardless of the number of processes
    - $\forall n. (P^n | L) \models \text{MutEx}$

- **Specification**
  - Mutual Exclusion (MutEx):
  - At most one process in $C$

Infinite-State System

$P^n | L$
Background
Parameterized Systems
Petri Nets
Lossy Channel Systems
Timed Petri Nets
Petri Nets

- Configurations
- Transitions

- Model

- Ordering

- Monotonicity
- Computing Predecessors
- Backward Reachability
- Upward Closed Sets
Petri Nets

Model

Configurations

Transitions

Ordering

Monotonicity

Upward Closed Sets

Computing Predecessors

Backward Reachability
Petri Nets
Petri Nets
Petri Nets

\begin{itemize}
\item \textit{places}
\item \textit{transitions}
\end{itemize}
Petri Nets

- Model
- Configurations
- Monotonicity
- Backward Reachability
- Transitions
- Ordering

- Upward Closed Sets
- Computing Predecessors
- Backward Reachability
Model

Configurations

Ordering

Transitions

Monotonicity

Upward Closed Sets

Computing Predecessors

Backward Reachability

Petri Nets
Markings
Markings

Petri Nets
Petri Nets

Markings

$t_1$ $t_2$

marking
Markings

Petri Nets

$t_1$  $t_2$

marking  multiset
Petri Nets

Transitions

$\overrightarrow{t_1} \overrightarrow{t_2}$
Transitions

\[ t_1 \quad t_2 \]

transition

\[ t_1 \]
Petri Nets

Transitions

transition

$t_1$
Transitions

Petri Nets

$t_1$ → transition → $t_2$
Petri Nets
Transitions
Petri Nets

Transitions

$t_1$ $t_2$

transition

$t_1$
Petri Nets

Transitions

$t_1 \quad t_2$

transition

$t_1$
Transitions
Petri Nets

Modeling

- Encoding (counter abstraction)
  - # tokens in \( L \) = # processes in \( L = \text{free?} \)
  - # tokens in \( I \) = # processes in \( L = \text{busy} \)
  - one/no token in \( C \) = lock free/busy

\( t_1 \quad I \quad t_2 \quad L \quad C \)

\( I \quad L := \text{free?} \quad L := \text{free} \quad L := \text{busy} \quad C \)
Petri Nets

Transitions

\[ t_1 \rightarrow t_2 \]
Petri Nets

Transitions

\[ t_1 \rightarrow t_2 \rightarrow t_1 \]
Petri Nets

Transitions

$t_1$
Petri Nets

Transitions

$t_1$
Petri Nets

Transitions

$t_1$ $t_2$
Petri Nets
Petri Nets

Transitions

\[ t_1 \quad t_2 \]
Petri Nets
Transitions

\[ t_1 \quad \text{black} \quad t_2 \]
Petri Nets

Transitions

$t_1$  $t_2$
Petri Nets
Transitions

$t_1$ $t_2$

$\text{Transitions}$
Petri Nets

Transitions

$t_1$

$t_2$

$t_1$

$t_2$
Petri Nets

Transitions

$\mathbf{t_1}$

$\mathbf{t_2}$
Petri Nets

Transitions

$t_1$ $t_2$

$t_1$

$t_2$

$t_1$
Petri Nets

Transitions

\[ t_1 \quad t_2 \]

\[ t_1 \quad t_2 \]

\[ t_1 \quad t_2 \]

\[ t_1 \quad t_2 \]

\[ t_1 \quad t_2 \]
Petri Nets

Transitions

```
\[
\begin{array}{c}
\text{t}_1 & \rightarrow & \text{t}_2 \\
\text{t}_2 & \rightarrow & \text{t}_1
\end{array}
\]
```
Safety Properties
Safety Properties

Initial Markings (Init)

- one
- arbitrarily many
Safety Properties

Initial Markings (\textit{Init})

- one
- arbitrarily many

\begin{itemize}
  \item one
  \item arbitrarily many
\end{itemize}
Safety Properties

Initial Markings (Init)
- infinitely many
- one
- arbitrarily many

Bad Markings (Bad)
- at least two
Safety Properties

Initial Markings (Init)
- infinitely many
- one
- arbitrarily many

Bad Markings (Bad)
- infinitely many
- at least two
Petri Nets

Safety Properties

Init

Bad

Initial Markings (Init)
- infinitely many
- one
- arbitrarily many

Bad Markings (Bad)
- infinitely many
- at least two
Petri Nets

Safety Properties

Initial Markings (Init)
- infinitely many
- one
- arbitrarily many

Bad Markings (Bad)
- infinitely many
- at least two

Safety Property

Init $\xrightarrow{*}$ Bad ?
Safety Properties

Init Markings (Init)
- infinitely many
- one
- arbitrarily many

Bad Markings (Bad)
- infinitely many

Safety Property
Init $\rightarrow^* \text{Bad}$?
**Petri Nets**

**Safety Properties**

**Initial Markings** (*Init*)
- infinitely many
- one
- arbitrarily many

**Bad Markings** (*Bad*)
- at least two
- infinitely many

**Safety Property**

*Init* $\rightarrow^*$ *Bad* ?
Petri Nets

Safety Properties

Initial Markings (Init)
- infinitely many
- one
- arbitrarily many

Bad Markings (Bad)
- infinitely many
- at least two

Safety Property

Init \* Bad ?
Safety Properties

How to check safety properties?

Init Markings (Init)
- infinitely many
- one
- arbitrarily many

Bad Markings (Bad)
- infinitely many
- at least two

Safety Property

$$Init \rightarrow^* Bad$$
Safety Properties

Initial Markings (Init)
- infinitely many

Bad Markings (Bad)
- at least two
  - one
  - arbitrarily many

How to check safety properties?
- Ordering
- Monotonicity
- Upward Closed sets
- Predecessors
- Backward Reachability

Safety Property

```
Init * Bad
```
Petri Nets

- Model
- Configurations
- Ordering
  - Monotonicity
  - Upward Closed Sets
  - Backward Reachability

- Transitions
  - Computing Predecessors
Petri Nets

- Model
  - Configurations
  - Transitions
- Ordering
  - Monotonicity
    - Upward Closed Sets
    - Computing Predecessors
    - Backward Reachability
Petri Nets

Ordering

\[ t_1 \quad t_2 \]
Petri Nets

Ordering

$\mathcal{P} = \{t_1, t_2\}$

$\mathcal{N} = \{1, 2, 3, 4, 5, 6\}$

$\mathcal{E} = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1)\}$

$\mathcal{F} = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1)\}$

$\mathcal{P} \cap \mathcal{N} = \emptyset$

$\mathcal{P} \cup \mathcal{N} = \{1, 2, 3, 4, 5, 6, t_1, t_2\}$

$\mathcal{P} \times \mathcal{N} = \{(t_1, 1), (t_1, 2), (t_1, 3), (t_1, 4), (t_1, 5), (t_1, 6), (t_2, 1), (t_2, 2), (t_2, 3), (t_2, 4), (t_2, 5), (t_2, 6)\}$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$

$\mathcal{E} \cap \mathcal{N} \times \mathcal{P} = \emptyset$

$\mathcal{E} \cap \mathcal{P} \times \mathcal{N} = \emptyset$
Petri Nets

- Model
- Configurations
- Transitions
- Ordering
- Monotonicity
- Upward Closed Sets
- Computing Predecessors
- Backward Reachability
Petri Nets

Monotonicity

Monotonicity
Petri Nets

Monotonicity

Monotonicity
Petri Nets

Monotonicity
Monotonicity
Petri Nets

Monotonicity

Monotonicity
Petri Nets

Monotonicity

\[ t_1 \quad t_2 \]

\[ t_1 \quad t_2 \]

Monotonicity
Monotonicity
Monotonicity

larger configuration
“simulate”
smaller configurations
Petri Nets

- Configurations
- Transitions
- Ordering
- Monotonicity
- Upward Closed Sets
- Computing Predecessors
- Backward Reachability

Model

✔️
Petri Nets

- Model
  - Configurations
  - Ordering
    - Monotonicity
      - Upward Closed Sets
      - Backward Reachability
  - Transitions
    - Computing Predecessors
Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \subseteq m_2$
- then $m_2 \in U$
Upward-Closed Set

Upward Closed Set (UC)

- if \( m_1 \in U \) and \( m_1 \sqsubseteq m_2 \)
- then \( m_2 \in U \)
Upward-Closed Set

Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \subseteq m_2$
- then $m_2 \in U$
Petri Nets

Upward Closed Sets

Upward-Closed Set

Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \sqsubseteq m_2$
- then $m_2 \in U$
Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \subseteq m_2$
- then $m_2 \in U$
Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \subseteq m_2$
- then $m_2 \in U$
Upward-Closed Set

Upward Closed Set (UC)
- if $m_1 \in U$ and $m_1 \subseteq m_2$
- then $m_2 \in U$
Upward Closed Set (UC)

- if $m_1 \in U$ and $m_1 \subseteq m_2$
- then $m_2 \in U$
Upward Closed Set (UC)

- if \( m_1 \in U \) and \( m_1 \subseteq m_2 \)
- then \( m_2 \in U \)

Why UC?

- Bad sets of markings are UC
  - checking safety properties = reachability of bad markings
- Uniquely characterized by generator
  - simple representation = finite multiset
Petri Nets
Upward Closed Sets
Petri Nets

Upward Closed Sets

\[ \text{implies} \]
Petri Nets

Upward Closed Sets

implies

generator
Petri Nets
Upward Closed Sets

implies

generator

generator
Petri Nets
Upward Closed Sets

implies

generator

\( \subseteq \)

\( \subseteq \)

generator
Petri Nets

- Model
- Configurations
- Transitions
- Ordering
- Monotonicity
- Upward Closed Sets
- Computing Predecessors
- Backward Reachability
Petri Nets

Model

Configurations

Ordering

Transitions

Monotonicity

Upward Closed Sets

Computing Predecessors

Backward Reachability
Petri Nets

Predecessors

$U$

$m_1$

$m_3$
Petri Nets

Predecessors

$\text{Pre}(U)$

$U$

$m_2$

$m_1$

$m_3$
Monotonicity: UC persevered by $Pre$

\[ m_1 \xrightarrow{\text{Pre}} m_2 \]
\[ m_3 \xrightarrow{\exists} m_4 \]
Petri Nets

Monotonicity: UC persevered by $\text{Pre}$

$m_1 \rightarrow m_2$

$m_3 \rightarrow m_4$

$\text{Pre}(U)$

$\uparrow$

$\uparrow$

$\exists$

$\uparrow$

$U$

upward closed

upward closed?
Petri Nets

Predecessors

Monotonicity: UC persevered by $Pre$

$m_1 \xrightarrow{\sqcap} m_2$

$m_3 \xrightarrow{\exists} m_4$

$Pre(U)$ upward closed?

$U$ upward closed
Monotonicity: UC persevered by $\text{Pre}$

$\text{Pre}(U)$ upward closed?

$U$ upward closed
Monotonicity: UC persevered by $Pre$

$Pre(U)$

upward closed?
Petri Nets

Predecessors

Monotonicity: UC persevered by \( \text{Pre} \)

\[
\begin{align*}
&m_1 \quad m_2 \\
&m_3 \quad m_4
\end{align*}
\]

\[
\begin{align*}
\text{Pre}(U) & \quad \text{upward closed?}
\end{align*}
\]

\( U \)

upward closed
Monotonicity: UC persevered by $Pre$

$Pre(U)$

$U$

$m_1 \rightarrow m_2$

$m_3 \rightarrow m_4$

$\forall m, m \in Pre(U)$

$\exists m' \in U$ such that $m \rightarrow m'$
Computing Predecessors

$Pre_{t_1} =$

Petri Net Diagram:

- Transitions $t_1$ and $t_2$
- Tokens in places
- Arrows indicating the flow of tokens
Petri Nets
Computing Predecessors

\[ Pre_{t_1} \]
Computing Predecessors

\[ \text{Pre}_{t_1} \begin{array} {c} \text{ } \\ \text{ } \\ \text{ } \end{array} \begin{array} {c} \text{ } \\ \text{ } \\ \text{ } \end{array} = \begin{array} {c} \text{ } \\ \text{ } \\ \text{ } \end{array} \]

\[ \text{Pre}_{t_2} \begin{array} {c} \text{ } \\ \text{ } \\ \text{ } \end{array} \begin{array} {c} \text{ } \\ \text{ } \\ \text{ } \end{array} = \begin{array} {c} \text{ } \\ \text{ } \\ \text{ } \end{array} \]
Petri Nets

Computing Predecessors

\[ \text{Pre}_{t_1} \begin{bmatrix} \text{red} \end{bmatrix} = \begin{bmatrix} \text{red}, \text{blue} \end{bmatrix} \]

\[ \text{Pre}_{t_2} \begin{bmatrix} \text{red} \end{bmatrix} = \begin{bmatrix} \text{red} \end{bmatrix} \]
Computing Predecessors

\[ \text{Pre}_{t_1} = \]

\[ \text{Pre}_{t_2} = \]

\[ \text{Pre}_{t_1} = \]
Petri Nets
Computing Predecessors

$Pre_{t_1}[[\text{red}]] = \text{green} + \text{blue}$

$Pre_{t_2}[[\text{red}]] = \text{green}$

$Pre_{t_1}[[\text{green} + \text{blue}]] = \text{green} + \text{blue}$
Petri Nets

- Model
  - Configurations
    - Ordering
      - Monotonicity
        - Upward Closed Sets
        - Backward Reachability
        - Computing Predecessors
  - Transitions
Petri Nets

- Model
- Configurations
- Transitions

Ordering

- Monotonicity
- Upward Closed Sets
- Computing Predecessors

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets
Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability

\[ t_1 \quad t_2 \]
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability

Diagram showing a Petri net with transitions $t_1$ and $t_2$. The net has places and transitions connected by arcs.
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability

\[
t_1 \quad t_2
\]
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets
Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability

Diagram:

- States: $t_1$, $t_2$
- Transitions: $t_2$ connects $t_1$ and $t_1$ connects $t_2$

Diagram elements:
- Blue circle
- Green circle
- Red circle
- Black squares

Notes:
- Petri Nets are used to model systems with concurrent processes.
- Backward reachability helps in understanding the predecessors of a state in the system.
Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability

Diagram: A Petri net with places $t_1$ and $t_2$ connected by transitions.
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability

\[ t_1 \rightarrow t_2 \]
Backward Reachability

Petri Nets
Petri Nets

Backward Reachability
Petri Nets

Backward Reachability

$Pre^*(\cdot)$
Petri Nets

Backward Reachability

$Pre^*(\cdot)$
Petri Nets

Backward Reachability

System Safe!

$Pre^*(\text{ })$
Backward Reachability

System Safe!

symbolic representation = finite multisets

\[ \text{Pre}^*(\text{initial markings}) \]
Petri Nets

Backward Reachability

System Safe!

Initial markings

Symbolic representation = finite multisets

Termination: multisets well quasi-ordered
Well Quasi-Ordering

$m_0, m_1, m_2, \ldots, m_i, \ldots, m_j, \ldots$

infinite sequence of markings
Well Quasi-Ordering

$m_0, m_1, m_2, \ldots, m_i, \ldots, m_j, \ldots$

$\exists i < j : m_i \sqsubseteq m_j$
Petri Nets

Well Quasi-Ordering

Well Quasi-Ordering

\[ m_0, m_1, m_2, \ldots, m_i, \ldots, m_j, \ldots \]

\[ \exists i < j : m_i \preceq m_j \]
Assume: non-termination
Petri Nets

Backward Reachability

Termination

Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: 
non-termination
Assume: non-termination

Petri Nets

Backward Reachability

Termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination
Assume: non-termination

contradiction: WQO
Petri Nets
Backward Reachability

System Safe!

Symbolic representation = finite multisets

Termination: multisets well quasi-ordered

Initial markings
Petri Nets

Backward Reachability

Initial markings

Symbolic representation = finite multisets

System Safe!

Termination: multisets well quasi-ordered

Ordering:
  • monotonicity
  • computing predecessors
  • well quasi-ordering
Petri Nets

\[ p_1 \rightarrow t_1 \rightarrow p_2 \leftarrow t_2 \rightarrow p_3 \leftarrow t_3 \]
• Perform backward reachability analysis from \([p3,p3]\)
• Reachable from:
  • \([p1,p1]\)?