Well-Structured Transition Systems and Extended Petri Nets
—An Introduction—

Jean-François Raskin
ULB

AVACS Spring School - Oldenburg - March 2010
Plan of the talk

- Parametric systems - Parametric verification
- Well-quasi orders and well-structured transition systems
- Extended Petri nets
- Three algorithmic tools for WSTS:
  - The set saturation method
  - The finite unfolding (≠ “Karp-Miller” tree)
  - The “Expand, Enlarge and Check” (EEC) algorithm
- Beyond this introduction - bibliography
- Conclusion
Introduction
Motivations

- Protocols are often designed to work for an arbitrary number of participants
- Multi-threaded programs may trigger the creation of an unbounded number of threads
- We need abstract models to reason about such systems
- We need techniques to establish correctness for an arbitrary number of participants/threads...
- We want parametric verification!
Parametric verification and PN

mutex M;

Process P {
    repeat {
        take M;
        critical;
        release M;
    }
}
Parametric verification and PN

mutex M;

Process P {
    repeat {
        take M;
        critical;
        release M;
    }
}
Mutex M;

Process P {
    repeat {
        take M;
        critical;
        release M;
    }
}

Mutual exclusion is verified if there is no more than one token in the red place in any reachable marking.
Motivations

• Protocols are often designed to work for an arbitrary number of participants

• Multi-threaded programs may trigger the creation of an unbounded number of threads

• We need abstract models to reason about such protocols/programs.

  • Well structured transition systems (WSTS) are such abstract models.

  • WSTS enjoy general decidability results.
Parametric verification and PN

mutex M;

Process P {
    repeat {
        take M;
        critical;
        release M;
    }
}

Mutual exclusion token

This is a **coverability** property!

Coverability properties are decidable for the class of WSTS!
Well quasi-orders
Well Structured Transition Systems
Well quasi-order

• Let $S$ be a (possibly infinite) set, a relation $\leq \subseteq S \times S$ is
  • A pre-order iff $\leq$ is reflexive and transitive;
  • A partial-order iff $\leq$ is a pre-order and antisymmetric;
  • A total order iff $\leq$ is a partial-order and total.

• $(S, \leq)$ is an ordered set if $\leq$ is a pre-order on $S$. 
Well quasi-order

- Let \((S, \leq)\) be an ordered set, \(\leq\) is well-founded iff there is no infinite decreasing chains.

\[
S_1 > S_2 > S_3 > ... > S_n > ....
\]

- Let \((S, \leq)\) be an ordered set, \(\leq\) is a well-quasi ordering (WQO) iff in any infinite sequence \(S_1 S_2 ... S_i ...\) there exist two positions \(k < l\) s.t. \(S_k \leq S_l\).

\[
S_1 \ S_2 \ ... \ S_k \ ... \ S_l \ ...
\]
Well quasi-order

- $(S, \leq)$ is called a well-quasi ordered set if $\leq$ is a WQO.
- Clearly, all well-quasi ordered sets $(S, \leq)$ are well-founded sets.
- The set $(\mathbb{N}, \leq)$ is a well-quasi ordered set.
The set \((\mathbb{N}, \leq)\) is a well-quasi ordered set

Indeed, consider for the sake of contradiction that it is not the case.

Then there exists a sequence of natural numbers \(n_0 n_1 ... n_i ...\) such that for all \(k<l : \neg(n_k \leq n_l)\).

But as \(\leq\) is a total order, we have then for all \(k<l : n_k>n_l\) i.e., an infinite strictly decreasing sequence of elements which is not possible.
**Lemma.** Let \((S, \leq)\) be a WQO set. From every infinite sequence \(s_1 s_2 \ldots s_j \ldots \) in \(S\) we can extract an infinite subsequence which is increasing i.e., a subsequence \(s_{f(1)} s_{f(2)} \ldots s_{f(j)} \ldots \) with \(f(i) < f(i+1)\) for all \(i \geq 1\), and such that \(s_{f(i)} \leq s_{f(i+1)}\) for all \(i \geq 1\).

from

\[
S_1 \ S_2 \ S_3 \ \ldots \ S_n \ \ldots
\]

we can extract

\[
s_{f(1)} \leq s_{f(2)} \leq \ldots \leq s_{f(i)} \leq \ldots
\]

with

\[
f(1) < f(2) < \ldots < f(i) < \ldots
\]
\( (\mathbb{N}^k, \preceq) \) is a well quasi-ordered set

- The set \( (\mathbb{N}^k, \preceq) \), where \( \preceq \) is the pointwise extension of \( \leq \) on \( k \)-tuples of natural number i.e.,

  \[(c_1, c_2, \ldots, c_k) \preceq (d_1, d_2, \ldots, d_k) \iff c_i \leq d_i \text{ for all } i, 1 \leq i \leq k.\]

- .... is a well-quasi ordered set.
\((\mathbb{N}^k, \preceq)\) is a well quasi-ordered set

By induction on \(k\). If \(k=1\), the theorem holds as \((\mathbb{N}, \leq)\) is a well-quasi ordered set.

**Induction.** Let \(k=i>1\). By induction hyp. \((\mathbb{N}^{k-1}, \preceq)\) is WQO set.

Assume for the sake of contradiction that \(v_1v_2...v_j...\) is an infinite sequence of incomparable elements in \((\mathbb{N}^k, \preceq)\).

Let us consider the projection of this sequence on the dimensions 2,3,...,\(k\) : \(v_1(2..i) v_2(2..i)...v_j(2..i)...

By induction hypothesis \((\mathbb{N}^{k-1}, \preceq)\) is WQO and so we can extract an infinite subsequence of increasing elements in \(\mathbb{N}^{k-1}\). Let \(f(1)f(2)...f(j)\)... be the indices corresponding to this subsequence.

Clearly the sequence \(v_{f(1)}(1)v_{f(2)}(1)...v_{f(j)}(1)\)... must be a sequence of pairwise incomparable elements. But this contradict the fact that \((\mathbb{N}, \leq)\) is a WQO set.
Upward and downward closed sets

- Let $(S, \leq)$ be an ordered set.
- The set $U \subseteq S$ is **upward-closed**
  iff for all $u \in U$ for all $s \in S$ : if $u \leq s$ then $s \in U$.
- The set $D \subseteq S$ is **downward-closed**
  iff for all $d \in D$ for all $s \in S$ : if $s \leq d$ then $s \in D$.
Upward and downward closed sets

- Let \((S, \leq)\) be a ordered set.

- Let \(S' \subseteq S\). The **upward-closure** of \(S'\), noted \(\uparrow S'\), is the set \(\{ s \in S \mid \exists s' \in S' \cdot s' \leq s \}\).

- Let \(S' \subseteq S\). The **downward-closure** of \(S'\), noted \(\downarrow S'\), is the set \(\{ s \in S \mid \exists s' \in S' \cdot s \leq s' \}\).
Generators of upward closed sets

- Let \((S,\leq)\) be a ordered set.
- A set \(A \subseteq S\) is an antichain if for all \(a_1, a_2 \in A\), if \(a_1 \neq a_2\) then neither \(a_1 \leq a_2\) nor \(a_2 \leq a_1\), i.e., \(a_1\) and \(a_2\) are incomparable.
- Let \(U \subseteq S\) be an upward closed set. A set \(G\) is a generator for \(U\) if \(\uparrow G = U\).
- Let \(U \subseteq S\) be an upward closed set. Then \(UGen(U)\) is a set of elements of \(S\) such that:
  - \(UGen(U) \subseteq U\);
  - \(UGen(U)\) is a generator for \(U\);
  - \(UGen(U)\) is an antichain.
Generators of upward closed sets

Let $U \subseteq S$ be an upward closed set. Then $\text{UGen}(U)$ is a set of elements of $S$ such that:

- $\text{UGen}(U) \subseteq U$;
- $\text{UGen}(U)$ is a generator for $U$;
- $\text{UGen}(U)$ is an antichain.
Generators of upward closed sets

**Theorem.** Let \((S, \leq)\) be a WQO. Let \(U \subseteq S\) be an upward closed set. Then there exists a set \(A \subseteq U\):

- \(A\) is an antichain;
- \(A\) is a generator of \(U\);
- \(A\) is finite.
Theorem. Let \((S, \leq)\) be a WQO. Let \(U\) be an upward closed set. Then there exists a set \(A \subseteq U:\)

- \(A\) is an antichain;
- \(A\) is a generator of \(U\);
- \(A\) is finite.

Generators of upward closed sets

\[ U = \text{UGen}(U) \]

If \(\leq\) is a partial order: take the finite set of minimal elements!
Theorem. Let $(S, \leq)$ be a WQO. Let $U \subseteq S$ be an upward closed set. Then there exists a set $A \subseteq U$:

- $A$ is an antichain;
- $A$ is a generator of $U$;
- $A$ is finite.

Generators of upward closed sets

If $\leq$ is a pre-order: take a representative in each equivalence class of minimal elements!

If $\leq$ is a partial order: take the finite set of minimal elements!
Upward closed sets in $(\mathbb{N}^k, \preceq)$

\[ \text{Min}(U) = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\} \] is a finite generator for $U$. 
Well Structured Transition Systems
A transition system is a tuple $T = (C, c_0, \rightarrow)$ where:

- $C$ is a (possibly infinite) set of configurations
- $c_0 \in C$ is the initial configuration
- $\rightarrow \subseteq C \times C$ is the transition relation
A well-structured transition system is a tuple $T=(C,c_0,\rightarrow,\leq)$ where:

- $(C,c_0,\rightarrow)$ is a transition system
- $(C,\leq)$ is a well-quasi ordered set
- $\rightarrow$ is monotonic: for all $c_1,c_2,c_3 \in C$:
  - if $c_1 \rightarrow c_2$ and $c_1 \leq c_3$ then there exists $c_4$: $c_3 \rightarrow c_4$ and $c_2 \leq c_4$. 
Well structured transition system

• A well-structured transition system is a tuple $T = (C, c_0, \rightarrow, \leq)$ where:
  
  • $(C, c_0, \rightarrow)$ is a transition system
  
  • $(C, \leq)$ is a well-quasi ordered set
  
  • $\rightarrow$ is monotonic: for all $c_1, c_2, c_3 \in C$:
    
    if $c_1 \rightarrow c_2$ and $c_1 \leq c_3$
    
    then there exists $c_4$: $c_3 \rightarrow c_4$ and $c_2 \leq c_4$. 

\[
\begin{array}{c}
\forall \\
\forall \\
C_1 & \rightarrow & C_2
\end{array}
\]
Well structured transition system

• A well-structured transition system is a tuple $T=(C,c_0,\rightarrow,\leq)$ where:
  
  • $(C,c_0,\rightarrow)$ is a transition system
  
  • $(C,\leq)$ is a well-quasi ordered set
  
  • $\rightarrow$ is monotonic: for all $c_1,c_2,c_3 \in C$:
    
    if $c_1 \rightarrow c_2$ and $c_1 \leq c_3$
    
    then there exists $c_4$: $c_3 \rightarrow c_4$ and $c_2 \leq c_4$. 

\[ \begin{align*}
C_3 & \quad \rightarrow \quad C_4 \\
\forall \quad \exists \quad \forall \quad \exists \\
C_1 & \quad \rightarrow \quad C_2
\end{align*} \]
Predicate transformer for TS

- **Predicate transformers:**
  - \( \text{Post}(c) = \{ c' | c \rightarrow c' \} \)
  - As usual, for \( S \subseteq C \), we write \( \text{Post}(S) = \bigcup_{c \in S} \text{Post}(c) \).
  - \( \text{Post}^1 = \text{Post} \) and \( \text{Post}^i = \text{Post} \circ \text{Post}^{i-1} \) and \( \text{Post}^* = \bigcup_{i \geq 0} \text{Post}^i \).
  - \( \text{Reach}(T) = \text{Post}^*(c_0) \).
  - \( \text{Pre}(c) = \{ c' | c' \leftarrow c \} \)
  - As usual, for \( S \subseteq C \), we write \( \text{Pre}(S) = \bigcup_{c \in S} \text{Pre}(c) \).
  - \( \text{Pre}^1 = \text{Pre} \) and \( \text{Pre}^i = \text{Pre} \circ \text{Pre}^{i-1} \) and \( \text{Pre}^* = \bigcup_{i \geq 0} \text{Pre}^i \).
Petri nets and Extended Petri nets
Petri nets are an important and traditional model for modeling concurrent systems.
Exemple of PN

\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, 2, 0, 1) \]

\[ m_2 = (1, 1, 1, 0) \]

\[ m_3 = (1, 2, 0, 1) \]

\[ m_4 = (1, 3, 0, 1) \]

\[ m_5 = (1, 2, 1, 0) \]

\[ m_6 = (1, 3, 0, 1) \]
Exemple of PN

$m_0 = (1, 1, 0, 1)$

$m_1 = (1, 2, 0, 1)$

$m_1 = (1, 1, 1, 0)$

$m_1 = (1, 2, 0, 1)$

$m_1 = (1, 3, 0, 1)$

$m_1 = (1, 3, 0, 1)$

$m_1 = (1, 3, 0, 1)$

$m_1 = (1, 2, 1, 0)$

$m_1 = (1, 3, 0, 1)$

Friday 19 March 2010
Exemple of PN

\[ m_0 = (1,1,0,1) \]
\[ m_1 = (1,2,0,1) \]
\[ m_2 = (1,2,1,0) \]
\[ m_3 = (1,3,0,1) \]

\[ t_1 \]
\[ t_2 \]
\[ t_3 \]

Friday 19 March 2010
Exemple of PN

\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, 2, 0, 1) \]

\[ m_2 = (1, 1, 1, 0) \]

\[ m_3 = (1, 2, 0, 1) \]

\[ m_4 = (1, 3, 0, 1) \]

\[ m_5 = (1, 2, 1, 0) \]

\[ m_6 = (1, 3, 0, 1) \]

\[ \ldots \]

\[ \ldots \]

\[ \ldots \]
Extended Petri Nets

• A extended Petri net $N=(P,T,m_0)$ where :

  • $P=\{p_1,p_2,\ldots,p_n\}$ is a finite set of places;

  • $T=\{t_1,t_2,\ldots,t_m\}$ is a finite set of transitions, each of which is of the form $(I,O,s,d,b)$ where :

    ★ $I : P \rightarrow \mathbb{N}$ are multi-sets of input places, $I(p)$ represents the number of occurences of $p$ in $I$.

    ★ $O : P \rightarrow \mathbb{N}$ are multi-sets of output places.

    ★ $s,d \in P \cup \{\perp\}$ are the source and destination places of a special arc and $b \in \mathbb{N} \cup \{+\infty\}$ is the bound associated to the special arc.

• We partition $T$ into $T_r \cup T_e$ where $T_r$ contains regular transitions where $s=d=\perp$ and $b=0$, and $T_e$ contains extended transitions where $s,d \in P$ and $b \neq 0$. 
Extended Petri Nets

- A Petri net (PN) is a EPN where $T_e = \emptyset$.

- A Petri net with transfer arcs (PN+T) is such that for all $t=(I,O,s,d,b) \in T_e$, $b = +\infty$.

- A Petri net with non-blocking arcs (PN+NBA) is such that for all $t=(I,O,s,d,b) \in T_e$, $b = 1$.

- Extended Petri nets are useful to model synchronization mechanisms in counting abstractions such as non-blocking synchronization, broadcast, etc.
Example of PN+NBA
Example of PN+NBA

Non-blocking arcs

PN + NBA

At most one token gets moved from the source to the destination

Friday 19 March 2010
Example of PN+NBA

Non-blocking arcs

PN + NBA
At most one token gets moved from the source to the destination
Example of PN+NBA

Non-blocking arcs

PN + NBA

At most one token gets moved from the source to the destination
Example of PN+NBA

Non-blocking arcs

PN + NBA

At most one token gets moved from the source to the destination
Non-blocking arcs

PN + NBA

At most one token gets moved from the source to the destination
Example of PN+NBA

Non-blocking arcs

PN + NBA

At most one token gets moved from the source to the destination

Friday 19 March 2010
Example of PN+NBA

Non-blocking arcs

PN + NBA
At most one token gets moved from the source to the destination
Example of PN+NBA
Example of PN+NBA

$t_1$ can be fired in this marking
Example of PN+NBA

t₁ can be fired in this marking.
Firing t₁ removes one token in p₁, one token in s, add one token to p₂ and one token to d.
Example of PN+NBA

t_1 can be fired in this marking
Example of PN+NBA

$t_1$ can be fired in this marking

Firing $t_1$ removes one token in $p_1$, add one token to $p_2$. 
Example of PN+T
Example of PN+T

Transfer arcs

PN + T

All the tokens are moved from the source to the destination
Example of PN+T

Transfer arcs
PN + T
All the tokens are moved from the source to the destination
Example of PN+T

Transfer arcs

PN + T

All the tokens are moved from the source to the destination
Example of PN+T

Transfer arcs

PN + T

All the tokens are moved from the source to the destination
Example of PN+T

t₁ can be fired in this marking
Example of PN+T

$t_1$ can be fired in this marking

When firing $t_1$, one token is removed from $p_1$ and added to $p_2$, and all the tokens in $s$ are transferred to $d$. 
Semantics of PN

• Let $N=(P,T,m_0)$ be a Petri net.

• Its semantics is given by the following transition system $Tr(N)=(C,c_0,\rightarrow)$ where:
  
  • $C=\{ m \mid m : P \rightarrow \mathbb{N} \}$
  
  • $c_0=m_0$
  
  • for all $m_1,m_2 \in C$, $m_1 \rightarrow m_2$ iff there exists $t=(I,O) \in T$:
    
    • $I \leq m_1$ and
    
    • $m_2=m_1-I+O$.  

Friday 19 March 2010
Semantics of Extended Petri nets

- Let \( N=(P,T,m_0) \) be an extended Petri net.

- Its semantics is given by the following transition system
  \( \text{Tr}(N)=(C,c_0,\Rightarrow) \) where: \( C=\{ m \mid m : P \to \mathbb{N} \} \), \( c_0=m_0 \), and:

  - for all \( m,m' \in C \), \( m \Rightarrow m' \) iff there exists \( t=(I,O,s,d,b) \in T \) and \( I \leq m \), and \( m' \) is computed as follows: let \( m_1=m-I \)
    - Compute \( m_2 \) as follows: if \( s=d=\perp \) then \( m_2=m_1 \)
      otherwise \( m_2 \) agrees with \( m_1 \) on all places but \( s \) and \( d \) where:
        - \( m_2(s)=\max(0,m_1(s)-b) \)
        - \( m_2(d)=\min(m_1(d)+m_1(s),m_1(d)+b) \)
  - Finally \( m'=m_2+O \)
Let $N=(P,T,m_0)$ be an extended Petri net. Its transition system $\text{Tr}(N)=(C,c_0,\rightarrow)$ is a WSTS $(C,c_0,\rightarrow,\preceq)$, where:

- $\preceq$ is the extension of $\leq \subseteq \mathbb{N} \times \mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
- and $\rightarrow$ is monotonic w.r.t. $\preceq$. 

- EPN are WSTS
EPN are WSTS

Let $N=(P,T,m_0)$ be an extended Petri net. Its transition system $\text{Tr}(N)=(C,c_0,\rightarrow)$ is a WSTS $\text{WSTS}(C,c_0,\rightarrow,\preceq)$, where:

- $\preceq$ is the extension of $\leq \subseteq \mathbb{N} \times \mathbb{N}$ to tuples in $\mathbb{N}^{\mid P\mid}$, it is a WQO.
- and $\rightarrow$ is monotonic w.r.t. $\preceq$.

$m_1=(2,0,3,0)$
Let $N=(P,T,m_0)$ be an extended Petri net. Its transition system $\text{Tr}(N)=(C,c_0,\Rightarrow)$ is a WSTS $(C,c_0,\Rightarrow,\leq)$, where:

- $\leq$ is the extension of $\leq \subseteq \mathbb{N} \times \mathbb{N}$ to tuples in $\mathbb{N}^{\text{I}^P}$, it is a WQO.
- $\Rightarrow$ is monotonic w.r.t. $\leq$.

$m_1=(2,0,3,0) \rightarrow m_2=(1,1,2,1)$
Let $\mathcal{N}=(P,T,m_0)$ be an extended Petri net. Its transition system $\text{Tr}(\mathcal{N})=(C,c_0,\rightarrow)$ is a WSTS $(C,c_0,\rightarrow,\preceq)$, where:

- $\preceq$ is the extension of $\leq \subseteq \mathbb{N} \times \mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
- and $\rightarrow$ is monotonic w.r.t. $\preceq$.

$m_3=(3,0,4,0)$

$m_1=(2,0,3,0) \rightarrow m_2=(1,1,2,1)$

Friday 19 March 2010
EPN are WSTS

- Let $N=(P,T,m_0)$ be an extended Petri net. Its transition system $\text{Tr}(N)=(C,c_0,\rightarrow)$ is a WSTS $(C,c_0,\rightarrow,\preceq)$, where:
  - $\preceq$ is the extension of $\leq \subseteq \mathbb{N} \times \mathbb{N}$ to tuples in $\mathbb{N}^{|P|}$, it is a WQO.
  - and $\rightarrow$ is monotonic w.r.t. $\preceq$.

\[
\begin{align*}
 m_1 &= (2,0,3,0) \quad \rightarrow \quad m_2 = (1,1,2,1) \\
 m_3 &= (3,0,4,0) \quad \rightarrow \quad m_4 = (2,1,3,1)
\end{align*}
\]
Properties of extended Petri nets

- The **reachability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if $m \in \text{Post}^*(m_0)$.

- The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$.

- The **non-terminating computation problem** asks given a net $N=(P,T,m_0)$ if there exists an infinite computation in $N$ starting from $m_0$.

- The **place boundedness problem** asks given a net $N=(P,T,m_0)$ and a place $p \in P$ if there exists a bound $n \in \mathbb{N}$ such that for all $m \in \text{Reach}(m_0)$, we have that $m(p) \leq n$. 
Reachability is undecidable for EPN

**Theorem.** The reachability problem for PN+NBA (and for PN+T) is **undecidable.**
Theorem. The reachability problem for PN+NBA (and for PN+T) is undecidable.

Proof sketch. Given a 2CM machine $M$, we can construction a PN+NBA $N$ and two markings $m_0, m_1$ such that $m_1$ is reachable from $m_0$ in $N$ iff the machine $M$ halts.

We associate to each counter and each control state of the 2CM a place of the net. We have an additional place $p_{\text{check}}$.

Initially, the place associated to the initial control state contains one token, all the other places (including $p_{\text{check}}$ and the two counters) are empty.
**Theorem.** The reachability problem for PN+NBA (and for PN+T) is undecidable.

Simulation of the instructions of a 2CM.
Theorem. The reachability problem for PN+NBA (and for PN+T) is **undecidable**.

\[ L_1: c_1 := c_1 + 1; \text{goto } L_2. \]
Reachability is undecidable for EPN

**Theorem.** The reachability problem for PN+NBA (and for PN+T) is **undecidable**.

\[ L_1: \text{if } c_1 \neq 0 \text{ then } c_1 := c_1 - 1; \text{ goto } L_2 \text{ else goto } L_3. \]
**Theorem.** The reachability problem for PN+NBA (and for PN+T) is undecidable.

With this additional gadget, it is clear that the machine M halts \textbf{iff} the marking “one token in halt and all other places empty” is reachable for the initial marking.
Reachability is undecidable for EPN

**Theorem.** The reachability problem for PN+NBA (and for PN+T) is **undecidable**.

With this additional gadget, it is clear that the machine $M$ halts **iff** the marking “one token in halt and all other places empty” is reachable for the initial marking.
Place boundedness

**Theorem.** The place boundedness problems for PN+NBA and PN+T are undecidable.
Place boundedness

**Theorem.** The place boundedness problems for PN+NBA and PN+T are *undecidable*.

To prove that we need a non-trivial extension of the proof idea in the previous undecidability result.
Three algorithmic techniques for WSTS
Technique I: set saturation
Backward algorithm for coverability

- The coverability problem asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$. 
The coverability problem asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$. 
Backward algorithm for coverability

- The **coverability problem** asks given a net \( N = (P, T, m_0) \) and a marking \( m \), if there exists a marking \( m' \succeq m \) such that \( m' \in \text{Post}^*(m_0) \).
Backward algorithm for coverability

- The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m'\geq m$ such that $m'\in \text{Post}^*(m_0)$. 
Backward algorithm for coverability

- The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$.
The coverability problem asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$. 
The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$. 
The coverability problem asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \geq m$ such that $m' \in \text{Post}^*(m_0)$.
• **Lemma.** Let $T=(C,c_0,\rightarrow,\leq)$ be a WSTS and $U$ be an $\leq$-upward closed set of configurations in $T$. Pre($U$) is $\leq$-upward closed.
• **Lemma.** Let $T=(C, c_0, \rightarrow, \leq)$ be a WSTS and $U$ be an $\leq$-upward closed set of configurations in $T$. $\text{Pre}(U)$ is $\leq$-upward closed.

Proof. Let $c_1 \in \text{Pre}(U)$ and let us consider any $c_2$ such that $c_1 \leq c_2$.

We know that there exists $c_3 \in U$ and $c_1 \rightarrow c_3$.

By monotonicity, there exists $c_4$ such that $c_3 \leq c_4$ and $c_2 \rightarrow c_4$.

As $U$ is upward closed, we have that $c_4 \in U$ and so $c_2 \in \text{Pre}(U)$. 

![Diagram](attachment:image.png)
**Lemma.** Let $T = (C, c_0, \rightarrow, \leq)$ be a WSTS and $U$ be an $\leq$-upward closed set of configurations in $T$. $\text{Pre}(U)$ is $\leq$-upward closed.

Proof. Let $c_1 \in \text{Pre}(U)$ and let us consider any $c_2$ such that $c_1 \leq c_2$.

We know that there exists $c_3 \in U$ and $c_1 \rightarrow c_3$.

By monotonicity, there exists $c_4$ such that $c_3 \leq c_4$ and $c_2 \rightarrow c_4$.

As $U$ is upward closed, we have that $c_4 \in U$ and so $c_2 \in \text{Pre}(U)$. 

---

![Diagram](attachment:image.png)
Effective WSTS

- **PreUp(c)** is the set of all configurations whose one-step successors by $\rightarrow$ are larger or equal to $c$ i.e.:

  \[ \text{PreUp}(c) = \{ c' \mid \exists c'' : c' \rightarrow c'' \text{ and } c \leq c'' \} = \text{Pre}(\uparrow c) \]

- A WSTS $T=(C,c_0,\rightarrow,\leq)$ is **effective** (EWSTS) if:
  - given any pair of configurations $c_1$ and $c_2$ in $C$, one can decide if $c_1 \rightarrow c_2$ or not.
  - given any pair of configurations $c_1$ and $c_2$ in $C$, one can decide if $c_1 \leq c_2$ or not.
  - given any configuration $c \in C$, one can effectively compute $\text{UGen}(\text{PreUp}(c))$.

- If the set of successors $\text{Post}(c)$ of a configuration $c$ is finite and effectively computable, we say that the WSTS is **forward effective** (FEWSTS for short).
General backward for solving coverability in EWSTS

• Let $T=(C, c_0, \rightarrow, \leq)$ be EWSTS. Let $U \subseteq C$ be an upward closed set and $UGen(U)$ a finite generator for $U$.

• Consider now the sequence:
  
  $E_0 = UGen(U)$
  
  $E_i = UGen(PreUp(E_{i-1}) \cup \uparrow E_{i-1})$, for $i \geq 0$.

  • First, note that all elements of this sequence are computable as $T$ is an EWSTS.

  • Second, $\uparrow E_i$ is the set of configurations of $T$ that can reach a configuration in $U$ in $i$ steps or less.

  • Third, there exists a position $k \geq 0$ such that for all $l \geq k$, $\uparrow E_i = \uparrow E_k$. 
Assume that this is not the case.

Then, as the sequence $\uparrow E_i$ is increasing for $\subseteq$, there must exist a sequence of elements $e_1 \ e_2 \ ... \ e_n \ ...$

such that for all $i<j$, $\neg(e_i \leq e_j)$.

But this is in contradiction with the fact that $(S, \leq)$ is a well-quasi ordered set!
Let $T=(C,c_0,\rightarrow,\leq)$ be an EWSTS. Let $U \subseteq C$ be an upward closed set and $\text{UGen}(U)$ a finite generator for $U$.

Consider now the sequence:

$E_0 = \text{UGen}(U)$

$E_i = \text{UGen}(\text{PreUp}(E_{i-1}) \cup \uparrow E_{i-1})$, for $i \geq 0$.

First, note that all elements of this sequence are computable as $T$ is an EWSTS.

Second, $\uparrow E_i$ is the set of configurations of $T$ that can reach a configuration in $U$ in $i$ steps or less.

Third, there exists a position $k \geq 0$ such that for all $l \geq k$, $\uparrow E_l = \uparrow E_k$.

This sequence is thus an effective algorithm to decide coverability in EWSTS.
Decidability of coverability for EWSTS

**Theorem.** The coverability problem is decidable for EWSTS.
Backward algorithm for coverability

- The **coverability problem** asks given a net \( N=(P,T,m_0) \) and a marking \( m \), if there exists a marking \( m' \geq m \) such that \( m' \in \text{Post}^*(m_0) \).
The coverability problem asks given a net \( N=({P,T,m_0}) \) and a marking \( m \), if there exists a marking \( m' \geq m \) such that \( m' \in \text{Post}^*(m_0) \).
Backward algorithm for coverability

- The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$. 

$\text{Pre}^2(\uparrow m)$
The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in Post^*(m_0)$.

- $Pre^2(\uparrow m)$
- $m_4$
- $m_3$
- $m_1$
- $m$
The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$.
The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$.
Backward algorithm for coverability

- The **coverability problem** asks given a net $N=(P,T,m_0)$ and a marking $m$, if there exists a marking $m' \succeq m$ such that $m' \in \text{Post}^*(m_0)$.

After a finite number of iterations it stabilizes on a set of markings whose upward closure is equal to the set of markings that can reach a marking covering $m$.

$$= \text{Pre}^*(m)$$
Example

\[ \text{Pre}(\uparrow (0,0,1,1)) = ? \]

(0,0,1,1)
Example

\[(0,0,1,1)\]

\[(0,0,2,0)\]

\[(0,0,3,0)\]

\[(1,0,1,1)\]

\[\ldots\]
Example

\[
\text{UGen(Pre}(\uparrow m)) = \text{Min}\{ m' \in \mathbb{N}^{|P|} \mid m' \geq l(t) \land m' - l(t) + O(t) \geq m \} \quad (0,0,1,1)
\]
Example

\begin{align*}
\text{UGen(Pre(}\uparrow m\text{))} \\
= \text{Min}\{ m' \in \mathbb{N}^{\mid P\mid} \mid m' \geq l(t) \land m' - l(t) + O(t) \geq m \} \quad (0,0,1,1) \\
= \text{intersection of two upward-closed sets!}
\end{align*}
Example

For $t_3$

$$\text{UGen}(\text{Pre} (\uparrow m)) = \min \{ m' \in \mathbb{N}^{|P|} | m' \geq I(t) \land m' - l(t) + O(t) \geq m \}$$

$(0,0,1,0)$ $(0,0,2,-1)$

$(0,0,1,1)$
Example

For \( t_3 \)

\[
\begin{align*}
\text{UGen(Pre(↑m))} &= \min \{ m' \in \mathbb{N}_P^{|P|} | m' \geq l(t) \land m' - l(t) + O(t) \geq m \} \\
\end{align*}
\]
For $t_3$

\[ UGen(Pre(\uparrow m)) = \min\{ m' \in \mathbb{N}^{[P]} \mid m' \geq I(t) \land m' - I(t) + O(t) \geq m \} \]
$$\text{For } t_1$$

$$\text{UGen}(\text{Pre}(\uparrow m)) = \text{Min}\{ m' \in \mathbb{N}^{\left| P \right|} | m' \geq l(t) \land m' - l(t) + O(t) \geq m \}$$

(0,0,1,1)

(1,0,0,0) (1,-1,1,1)

Friday 19 March 2010
UGen(Pre(↑m))
=\text{Min}\{ m'\in\mathbb{N}^{\#P} | m'\geq l(t) \land m'-l(t)+O(t)\geq m \}

\( (0,0,1,1) \cap \bigcap_{t_1} (1,0,0,0) \cap (1,-1,1,1) \)
Example

\[ \text{UGen(Pre(} \uparrow m) \text{)} \]
\[ = \min \{ m' \in \mathbb{N}^{\lvert P \rvert} \mid m' \geq l(t) \land m' - l(t) + O(t) \geq m \} \]
Example

\[
\text{UGen}(\text{Pre}(\uparrow m)) \\
= \text{Min}\{ m' \in \mathbb{N}^{\left| P \right|} \mid m' \geqslant I(t) \land m' - I(t) + O(t) \geqslant m \} \\
= \text{Min}\{(1,0,1,1),(0,0,2,0),(0,1,0,1)\} \\
= \{(1,0,1,1),(0,0,2,0),(0,1,0,1)\}
\]
Example

\[
\text{UGen}(\text{Pre}(\uparrow m) \cup \uparrow m) = \text{Min}\{(1,0,1,1),(0,0,2,0),(0,1,0,1)\} \cup \uparrow\{(0,0,1,1)\} =\{(0,0,2,0),(0,1,0,1),(0,0,1,1)\}
\]

Friday 19 March 2010
Example

\[
\text{UGen}(\text{Pre}(\uparrow^m) \cup \uparrow^m) = \text{Min} \left( \{(1,0,1,1),(0,0,2,0),(0,1,0,1)\} \cup \{(0,0,1,1)\} \right) \quad (0,0,1,1)
\]
\[
= \{(0,0,2,0),(0,1,0,1),(0,0,1,1)\}
\]

\[\ldots\]
Set saturation methods for EPN

- **Theorem.** The coverability problem for extended Petri net is decidable.
Set saturation methods for EPN

- **Theorem.** The coverability problem for extended Petri net is decidable.

Nevertheless, the worst case complexity is high:

- **Theorem.** The coverability problem is ExpSpace-C for Petri nets.
- **Theorem.** The coverability problem is non-primitive recursive for transfer/reset/NBA PN.
Technique 2: Tree saturation
Tree saturation

Tree saturation =

Unfolding +

Rule to stop

**Objective**: construct a *finite* tree that represents (in some way) all the *computations* of the transition system.
Tree saturation for PN

\[ m_0 = (1,1,0,1) \]
\[ m_1 = (1,2,0,1) \]
\[ m_1 = (1,2,0,1) \]

Unfolding
Tree saturation for PN

\[ m_0 = (1,1,0,1) \]
\[ t_1 \]
\[ m_1 = (1,2,0,1) \]
\[ t_2 \]
\[ m_2 = (1,1,1,0) \]
\[ t_1 \]
\[ m_3 = (1,2,0,1) \]
\[ t_3 \]
\[ m_1 = (1,2,1,0) \]
\[ t_2 \]
\[ m_1 = (1,3,0,1) \]

Stop whenever we construct a marking with an ancestor which is \( \preceq \).

Friday 19 March 2010
Tree saturation for PN

\[ m_0 = (1, 1, 0, 1) \]
\[ m_1 = (1, 2, 0, 1) \]
\[ m_2 = (1, 1, 1, 0) \]
\[ m_3 = (1, 2, 0, 1) \]
\[ m_4 = (1, 3, 0, 1) \]

\[ t_1 \]
\[ t_2 \]
\[ t_3 \]

\[ \ldots \]
Tree saturation for PN

\[ m_0 = (1,1,0,1) \]

\[ m_1 = (1,2,0,1) \]
Tree saturation for PN

$m_0 = (1, 1, 0, 1)$
$t_1
m_1 = (1, 0, 1, 0)$

$(1, 2, 0, 1)$

Friday 19 March 2010
Tree saturation for PN

\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, 2, 0, 1) \quad (1, 0, 1, 0) \]

\[ m_1 = (1, 1, 1, 0) \]
Tree saturation for PN

$m_0 = (1,1,0,1)$
$t_1 \rightarrow (1,0,1,0)$
$m_1 = (1,2,0,1)$
$t_2 \rightarrow (1,0,1,0)$
$m_1 = (1,1,1,0)$
$t_3 \rightarrow (1,1,0,1)$
Tree saturation for PN

\[ m_0 = (1, 1, 0, 1) \]
\[ m_1 = (1, 2, 0, 1) \]
\[ m_1 = (1, 1, 1, 0) \]

We are done !!!
Tree saturation for FEWSTS

- The stopping rule of the tree saturation method is applicable to any FEWSTS.

Indeed, on every infinite branch of the unfolding, we are guaranteed that there exist a node annotated with a state that is larger than one of its ancestor! This is a direct consequence of WQO!

- So for every FEWSTS, there exists a finite tree, called the finite reachability tree, obtained by the tree saturation method:

Theorem. A finite reachability tree exists and is effectively computable for any FEWSTS.

(easy proof using WQO+König’s lemma)
Properties of the finite reachability

- Clearly the leafs of the $FRT(T)$ are nodes that either have no successors or contain a state which subsumes an ancestor. As a consequence, we have the following theorem.

- **Theorem.** $T=(C,c_0,\rightarrow\leq)$ has a non-terminating computation starting in $c_0$ iff $FRT(T)$ contains a subsumed node.
Properties of the finite reachability

- **Theorem.** $T = (C, c_0, \Rightarrow \leq)$ has a non-terminating computation starting in $c_0$ iff $\text{FRT}(T)$ contains a subsumed node.

\[(\Leftarrow)\]

and $c_1 \leq c_4$

Then clearly $c_0(c_1c_2c_3c_4)^\omega$ is a non-terminating computation in $T$
Properties of the finite reachability

• **Theorem.** $T = (C, c_0, \implies \leq)$ has a non-terminating computation starting in $c_0$ iff $FRT(T)$ contains a subsumed node.

\[
(\implies)
\]

Let $c_0 \ c_1 \ c_2 \ \ldots \ c_n \ \ldots$ be a non-terminating computation in $T$.

This computation has a prefix which labels a branch in $FRT(T)$.

This branch must end in a node that subsumes an ancestor (it cannot be a node with no successor).
The non-terminating computation problem

• **Theorem.** The *non-terminating computation problem* is **decidable** for the entire class of FEWSTS.
Karp and Miller tree for PN

- The **Finite Reachability Tree** should not be confused with The **Karp and Miller tree** for Petri Net.
- KM Tree = Unfolding + **Accelerations** + Stopping rules.
- KM Tree is an procedure for computing an effective representation of the set $\downarrow\text{Reach}(N)$ of a **Petri net** $N$. 
KM tree for PN

$m_0 = (1,1,0,1)$

$t_1$

$m_1 = (1,2,0,1)$
\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, \omega, 0, 1) \quad \text{Acceleration!} \]
\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, \omega, 0, 1) \]

\[ m_1 = (1, \omega, 1, 0) \]
KM tree for PN

\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, \omega, 0, 1) \]

\[ m_1 = (1, \omega, 1, 0) \]

\[ m_1 = (1, \omega, 0, 1) \]
KM tree for PN

\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, \omega, 0, 1) \]

\[ m_2 = (1, \omega, 1, 0) \]

\[ m_3 = (1, \omega, 0, 1) \] Stop!
KM tree for PN

\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, \omega, 0, 1) \]

\[ m_1 = (1, \omega, 1, 0) \]

\[ m_1 = (1, \omega, 0, 1) \]
KM tree for PN

$m_0 = (1, 1, 0, 1)$

$m_1 = (1, \omega, 0, 1)$

$m_1 = (1, 1, 0, 0)$

$m_1 = (1, \omega, 1, 0)$

$m_1 = (1, \omega, 0, 1)$

$p_1$ (t_1)

$p_2$ (t_2)

$p_3$ (t_3)

$p_4$ (t_3)
KM tree for PN

\[ m_0 = (1, 1, 0, 1) \]

\[ m_1 = (1, \omega, 0, 1) \]

\[ m_1 = (1, \omega, 1, 0) \]

\[ m_1 = (1, \omega, 0, 1) \]

Stop!
KM tree for PN

$m_0 = (1, 1, 0, 1)$

$t_1$  

$m_1 = (1, \omega, 0, 1)$  

$t_2$  

$(1, 0, 1, 0)$

$t_3$  

$m_1 = (1, \omega, 1, 0)$  

$t_2$  

$t_1$  

$(1, 0, 1, 0)$

$t_3$  

$m_1 = (1, \omega, 0, 1)$

$t_2$  

$t_3$  

$(1, 1, 0, 1)$

...
Karp and Miller tree for PN

- The Finite Reachability Tree should not be confused with The Karp and Miller tree for Petri Net.
- KM Tree = Unfolding + Accelerations + Stopping rules.
- KM Tree is a procedure for computing an effective representation of the set $\downarrow\text{Reach}(N)$ of a Petri net $N$.
- $\downarrow\text{Reach}(N)$ allows for deciding coverability:
  $$\exists m' \geq m \cdot m' \in \text{Post}^*(m_0) \iff m \in \downarrow\text{Reach}(N).$$
- $\downarrow\text{Reach}(N)$ allows for deciding place boundedness:
  $$\text{p is bounded in } N \iff \exists k \in \mathbb{N} \cdot \forall m \in \downarrow\text{Reach}(N) \cdot m(p) \leq k.$$
\(\omega\)-Markings and downward closed sets in \((\mathbb{N}^k, \preceq)\)

- A \(\omega\)-marking is a function \(m : P \to \mathbb{N} \cup \{\omega\}\).
- \(\omega\) = “any number of tokens”.
- A \(\omega\)-marking \(m\) represents a set of “plain” markings:

  Let \(m\) be an \(\omega\)-marking

  \[
  \downarrow m = \{ m' \in [P \to \mathbb{N}] | \forall p \in P : m'(p) \leq m(p) \}
  \]

- **Theorem.** For any downward-closed set of marking \(D\), there exists a finite set of \(\omega\)-marking \(M\) such that \(\downarrow M = D\).
Downward-closed sets in \((\mathbb{N}^k, \preceq)\)

\[\text{DGen}(D) = \{(x_1, y_1), (x_2, y_2), (\omega, y_3)\}\] is a finite generator for D.
Reach(N) is not constructible for EPN

- We have seen that:
  - Reach(N) is sufficient to decide place boundedness
  - Place boundedness is undecidable for EPN!
- So, Reach(N) is not computable for EPN!
Reach\( (N) \) is not constructible for EPN

- We have seen that:
  - \( \downarrow \text{Reach}(N) \) is sufficient to decide place boundedness
  - Place boundedness is \textbf{undecidable} for EPN!
- So, \( \downarrow \text{Reach}(N) \) is \textbf{not} computable for EPN!

Still, can we have a forward algorithm for coverability?
Expand-Enlarge and Check
Forward algorithm for coverability of WSTS

• We have just seen that \( \downarrow \text{Reach}(N) \) has always a finite representation but it is not effectively computable.

• Nevertheless, our solution for a forward algorithm for deciding coverability of EPN will rely on the existence of this finite representation.
Under-approx of $\downarrow \text{Reach}(S)$

- Let $N=(P,T,m_0)$ be an extended Petri net and $T(N)=(P \rightarrow \mathbb{N},m_0,\rightarrow,\preceq)$ its associated WSTS.

- Let $k \in \mathbb{N}$, and the two following families of finite sets:
  - $C_k$ be the set of markings $\{ m | m \in P \rightarrow [0..k] \} \cup \{m_0\}$
  - $L_k$ be the set of $\omega$-markings $\{ m | m \in P \rightarrow [0..k] \cup \{\omega\} \} \cup \{m_0\}$.

- $\text{UnderApprox}(N,k)=(C_k,m_0,\rightarrow_{\text{under}})$ where:
  - $\rightarrow_{\text{under}}=\rightarrow \cap C_k \times C_k$ i.e., transitions that leads to markings with more than $k$ tokens are discarded.

- **Lemma.** $\downarrow \text{Reach}(\text{UnderApprox}(N,k)) \subseteq \downarrow \text{Reach}(N)$. 
l'ensemble de marquage le réseau de Petri ci-contre a trois places, suivante :

\[ \text{An example} \]

\[ \begin{align*}
\langle 0, 1, 1 \rangle & \xrightarrow{t_1} \langle 2, 1, 1 \rangle \\
\langle 1, 2, 0 \rangle & \xrightarrow{t_2} \langle 1, 0, 2 \rangle \\
\langle 2, 1, 1 \rangle & \xrightarrow{t_3} \langle 1, 2, 0 \rangle \\
\end{align*} \]

\[ \text{Under}(\mathbb{N}, 2) \]
An example

Under(N,2)
Over-approx of Cover(S)

• We define $\text{Post}^k : L_k \rightarrow 2^{L_k}$ as follows:

$$\text{Post}^k(m) = \{m' \in L_k \mid m \Rightarrow_\omega m' \text{ or } \neg(m \Rightarrow_\omega m') \text{ and } \exists m'' \cdot m \Rightarrow_\omega m' : m' = \text{enlarge}(m'', k)\}$$

where $\text{enlarge}(m'', k)(p) = m''(p)$ if $m'(p) \leq k$
$$\omega \text{ otherwise}$$

• $\text{OverApprox}(N, k) = (L_k, m_0, \Rightarrow_{\text{over}})$ where:

• $(m_1, m_2) \in \Rightarrow_{\text{over}}$ iff $m_2 \in \text{Post}^k(m_1)$

• Lemma. $\downarrow \text{Reach}(N) \subseteq \downarrow \text{Reach}(\text{OverApprox}(N, k))$. 
An example

Un exemple de réseau de Petri.
k:=0;

Repeat:

“Expand”: Compute $D_{\text{Under}}:=\text{UnderApprox}(N,k)$

“Enlarge”: Compute $D_{\text{Over}}:=\text{OverApprox}(N,k)$

“Check” : if $D_{\text{Under}} \cap U \neq \emptyset$ return “positive”;
else if $D_{\text{Over}} \cap U = \emptyset$ return “negative”
else $k:=k+1$;
EEC Algorithm

\[ k := 0; \]

Repeat:

“Expand”: Compute \( D_{\text{Under}} := \text{UnderApprox}(N, k) \)

“Enlarge”: Compute \( D_{\text{Over}} := \text{OverApprox}(N, k) \)

“Check”: if \( D_{\text{Under}} \cap U \neq \emptyset \) return “positive”;
else if \( D_{\text{Over}} \cap U = \emptyset \) return “negative”
else \( k := k + 1; \)

Clearly this algorithm is sound as it uses:
- under-approximations to detect positive instances.
- over-approximations to detect negative instances.
**EEC Algorithm**

\[ k := 0; \]

**Repeat:**

**“Expand”:** Compute \[ D_{\text{Under}} := \text{UnderApprox}(N, k) \]

**“Enlarge”:** Compute \[ D_{\text{Over}} := \text{OverApprox}(N, k) \]

**“Check”:**

- If \[ D_{\text{Under}} \cap U \neq \emptyset \] return “positive”;
- Else if \[ D_{\text{Over}} \cap U = \emptyset \] return “negative”;
- Else \[ k := k + 1; \]

Clearly this algorithm is **sound** as it uses:
- under-approximations to detect **positive** instances.
- over-approximations to detect **negative** instances.

**But does it always terminate?**
Termination of EEC

• **Yes** it does always **terminate**!

• **Lemma (Positive instances)**. Let $m_0m_1...m_n$ be an execution that reaches $U$. Let $k$ be the maximal number of tokens in a place of a marking in this execution. Then $\text{UnderApprox}(N,k) \cap U \neq \emptyset$.

• **Lemma (Negative instances)**. Let $k = \max\{ m(p) \neq \omega \mid m \in \text{DGen}(\downarrow \text{Reach}(N))\}$. $\downarrow \text{Post}^{#k}(\downarrow \text{Reach}(N)) = \downarrow \text{Post}(\downarrow \text{Reach}(N))$, and so $\downarrow \text{OverApprox}(N,k) = \downarrow \text{Reach}(N)$. 
Beyond this introduction

Bibliography
Some interesting papers

- **General papers**
  
  
Some interesting papers

- More applications


Some interesting papers

• Relation with abstractions/Abstract interpretation/Domain theory:


  • Alain Finkel, Jean Goubault-Larrecq: Forward Analysis for WSTS, Part I: Completions. STACS 2009: 433-444

  • Alain Finkel, Jean Goubault-Larrecq: Forward Analysis for WSTS, Part II: Complete WSTS. ICALP (2) 2009: 188-199
Some interesting papers

- PhD Thesis:
Conclusion
Conclusion

- **Well-structured transition systems** are a general class of infinite state systems with **decidable** verification problems.

- They are useful to model:
  - parametric systems,
  - lossy channel systems,
  - broadcast protocols,
  - timed Petri nets,
  - complements of one-clock timed languages, etc.

- We have reviewed **three algorithmic tools** for their analysis.
Questions