Initiation à la Vérification

Stefan Schwoon

Winter 2011/12
Contact details

Name: Stefan Schwoon

Email: schwoon@lsv.ens-cachan.fr

Personal webpage: http://www.lsv.ens-cachan.fr/~schwoon

Course webpage: see MPRI wiki
Program for the second half

First half of the course:

CTL/CTL*/LTL, model-checking algorithms

for finite-state systems

Algorithms linear in number of states

Second half of the course:

More refined verification algorithms

Infinite-state model checking
State-space explosion

CTL*/LTL model checking linear in the size of the (low-level) transition system.

Good, but... the transition system may still be of very large size, in fact exponentially larger than the underlying high-level description.

Examples:

**Data:** TS with variables $v \in \mathcal{V}$ with domains $D_v$

$\Rightarrow$ low-level states $\prod_{v \in \mathcal{V}} D_v$

**Concurrency:** Parallel composition of components with states $S_i$

$\Rightarrow$ low-level states $\prod_{1 \leq n} S_i$

Phenomenon popularly called state-space explosion.
Fighting state-space explosion

Many techniques have been developed to alleviate this problem. We will look at some representative examples:

**Compression**: represent sets of states concisely, handle many states at once

**Reduction**: identify “equivalent” computations, examine only one representative

**Abstraction**: throw away information, obtaining a smaller system, then repair inaccuracies
To infinity and beyond

Infinite-state model checking: vast field, even more techniques.

Reasons for infinite states: data, control, unknown parameters

Problems with (un-)decidability

Will take a glimpse at two computation models:

- **Petri nets**: modelling of concurrency aspects
- **Pushdown systems**: sequential, adequate to model procedure calls
Petri nets
Literature about Petri Nets

For more in-depth coverage, a lot of literature about Petri nets is available, for instance:


Internet resources: Petri Nets World
http://www.informatik.uni-hamburg.de/TGI/PetriNets/
Petri nets

Petri nets are a basic model of parallel and distributed systems, designed by Carl Adam Petri in 1962 in his PhD Thesis. The basic idea is to describe state changes in a system with transitions.

Petri nets contain places (Stelle) and transitions (Transition) that may be connected by directed arcs.

Places symbolise states, conditions, or resources that need to be met/be available before an action can be carried out.

Transitions symbolise actions.
Behaviour of Petri nets

Places may contain tokens that may move to other places by executing (“firing”) actions.

A token on a place means that the corresponding condition is fulfilled or that a resource is available:

In the example, transition $t$ may “fire” if there are tokens on places $s_1$ and $s_3$. Firing $t$ will remove those tokens and place new tokens on $s_2$ and $s_4$. 
A Petri net is a tuple $N = \langle P, T, F, W, m_0 \rangle$, where

- $P$ is a finite set of places,
- $T$ is a finite set of transitions,
- the places $P$ and transitions $T$ are disjoint ($P \cap T = \emptyset$),
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation,
- $W : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}$ is the arc weight mapping (where $W(f) = 0$ for all $f \notin F$, and $W(f) > 0$ for all $f \in F$), and
- $m_0 : P \rightarrow \mathbb{N}$ is the initial marking representing the initial distribution of tokens.
Let $N = \langle P, T, F, W, m_0 \rangle$ be a Petri net. We associate with it the transition system $\mathcal{M} = \langle S, \Sigma, \Delta, I, AP, \ell \rangle$, where:

$$S = \{ m \mid m : P \rightarrow \mathbb{N} \}, \ I = \{ m_0 \}$$

$$\Sigma = T$$

$$\Delta = \{ (m, t, m') \mid \forall p \in P : m(p) \geq W(p, t) \land m'(p) = m(p) - W(p, t) + W(t, p) \}$$

$$AP = P, \ \ell(m) = \{ p \in P \mid m(p) > 0 \}$$

When $(m, t, m') \in \Delta$, we say that $t$ is enabled in $m$ and that its firing produces the successor marking $m'$; we also write $m \xrightarrow{t} m'$. 
Petri nets: Remarks

If \( \langle p, t \rangle \in F \) for a transition \( t \) and a place \( p \), then \( p \) is an input place of \( t \).

If \( \langle t, p \rangle \in F \) for a transition \( t \) and a place \( p \), then \( p \) is an output place of \( t \).

Let \( a \in P \cup T \). The set \( \bullet a = \{ a' \mid \langle a', a \rangle \in F \} \) is called the pre-set of \( a \), and the set \( a^\bullet = \{ a' \mid \langle a, a' \rangle \in F \} \) is its post-set.

When drawing a Petri net, we usually omit arc weights of 1. Also, we may either denote tokens on a place either by black circles, or by a number.
The place/transition net $\langle P, T, F, W, m_0 \rangle$ above is defined as follows:

- $P = \{p_1, p_2, p_3\}$,
- $T = \{t\}$,
- $F = \{\langle p_1, t \rangle, \langle p_2, t \rangle, \langle t, p_3 \rangle\}$,
- $W = \{\langle p_1, t \rangle \mapsto 2, \langle p_2, t \rangle \mapsto 1, \langle t, p_3 \rangle \mapsto 2\}$,
- $m_0 = \{p_1 \mapsto 2, p_2 \mapsto 5, p_3 \mapsto 0\}$. 
Bigger example: Dining philosophers

There are philosophers sitting around a round table.

There are forks on the table, one between each pair of philosophers.

The philosophers want to eat spaghetti from a large bowl in the center of the table.
Dining philosophers: Petri net
Synchronization by rendez-vous

Assume that we have a number of components with local actions and actions \( !m \) (send message \( m \)) and \( ?m \) (receive message \( m \)).

Transition into Petri net:

Places = union of local states

Transitions:

– for local actions \( (p, a, p') \) build a Petri transition \( t \) labelled with \( a \) and \( t^\bullet = \{p\}, t^\bullet = \{p'\} \);

– for pairs of actions \( (p, !m, p') \) and \( (q, ?m, q') \) build a Petri transition \( t \) labelled with \( m \) and \( t^\bullet = \{p, q\}, t^\bullet = \{p', q'\} \).

Similar translations possible for other models discussed in the course (asynchronous product, TS with variables, . . . )
Notation for markings

Often we will fix an order on the places (e.g., matching the place numbering), and write, e.g., $m_0 = \langle 2, 5, 0 \rangle$ instead.

When no place contains more than one token, markings are in fact sets, in which case we often use set notation and write instead $m_0 = \{p_5, p_7, p_8\}$.

Alternatively, we could denote a marking as a multiset, e.g. $m_0 = \{p_1, p_1, p_2, p_2, p_2, p_2, p_2\}$. 
Reachable markings

Let $m$ be a marking of a Petri net $N = \langle P, T, F, W, m_0 \rangle$.

The set of markings reachable from $m$ (the reachability set of $m$, written $\text{reach}(m)$), is the smallest set of markings such that:

1. $m \in \text{reach}(m)$, and

2. if $m' \xrightarrow{t} m''$ for some $t \in T$, $m' \in \text{reach}(m)$, then $m'' \in \text{reach}(m)$.

The set of reachable markings $\text{reach}(N)$ of a net $N = \langle P, T, F, W, m_0 \rangle$ is defined to be $\text{reach}(m_0)$.
Reachability Graph

Let $N = \langle P, T, F, W, m_0 \rangle$ be a Petri net with associated transition system $\mathcal{M} = \langle S, \Sigma, \Delta, I, AP, \ell \rangle$.

The reachability graph of $N$ is the rooted, directed graph $G = \langle S', \Delta', m_0 \rangle$, where $S'$ and $\Delta'$ are the restrictions of $S$ and $\Delta$ to $\text{reach}(N)$.

The reachability graph can be constructed in iterative fashion, starting with the initial marking and then adding, step for step, all reachable markings.
**$k$-Safeness**

**Definition:** Let $N$ be a net. If no reachable marking of $N$ can contain more than $k$ tokens in any place (where $k \geq 0$ is some constant), then $N$ is said to be $k$-safe.

**Example:** The following net is 1-safe.

![Diagram of a net](image)

Other example: the nets resulting from translating synchronous rendez-vous
A $k$-safe net has at most $(k + 1)^{|P|}$ reachable markings; for 1-safe nets, the limit is $2^{|P|}$.

In this case, there are finitely many reachable markings, and the construction of the reachability graph terminates.

On the other hand, if a net is not $k$-safe for any $k$, then there are infinitely many markings, and the construction will not terminate.

In the following, we will first consider 1-safe nets.
Reachability problem (for 1-safe nets)

Let $N$ be a Petri net and $m$ be a marking. The *reachability problem* for $N$, $m$ is to determine whether $m \in \text{reach}(N)$.

**Theorem:** The reachability problem for 1-safe Petri nets is PSPACE-complete.

**Proof:** (sketch)
upper bound: non-deterministically simulate net for at most $2^{|P|}$ steps;
hardness by reduction from QBF.

**Corollary:** Given a 1-safe net $N$ and a place $p$, it is PSPACE-complete to determine whether $\text{reach}(N)$ contains a marking $m$ such that $m(p) = 1$. 
The most straightforward way to solve the reachability problem on Petri nets is to construct the reachability graph. However, this can be very inefficient:

If there are \( n \) such components, then the reachability graph has size \( 2^n \).
In fact, the reachability graph does not take advantage of the concurrent nature of a Petri net.

We shall study a method more adapted to concurrent systems. It constructs a concise representation of the reachable markings. This representation is called unfolding.

Given the unfolding of $N$, it is NP-complete to determine whether a given marking is reachable in $N$. One can thus take advantage of the advances made in SAT-checking.
Petri nets: Unfoldings
Unfoldings are a data structure that represents the behaviour of a Petri net.

We will study it for 1-safe nets.

Unfoldings represent a trade-off in terms of time/space requirements; their size is in between that of a net and its reachability graph, and checking whether a marking is reachable becomes easier than for the net, but more difficult than from the reachability graph.
Unfoldings for finite transition systems

Let $\mathcal{T}$ be a finite transition system with initial state $X$. One can define the acyclic unfolding $U_\mathcal{T}$ (which is used for CTL model checking):

**Remark:** $U_\mathcal{T}$ can be viewed as a structure in which every state is labelled by a state from $\mathcal{T}$. We denote this labelling by the function $B$.

$U_\mathcal{T}$ contains the same behaviours as $\mathcal{T}$ (and the same reachable states). Additionally, $U_\mathcal{T}$ has a simpler structure (acyclic, in fact, a tree). However, in general, $U_\mathcal{T}$ is *infinite.*
$P$ is called a prefix of $U_T$ if $P$ is obtained by “pruning” arbitrary branches of $U_T$. Example:

Observation: One can always find a finite prefix containing the same reachable states as the infinite unfolding (by unrolling loops exactly once). We shall call such a prefix complete.
Construction of complete prefixes

Let us discuss an algorithm to obtain a complete prefix of $U_T$.

The algorithm maintains a set $\mathcal{E}$, the set of states observed so far.

Some arcs in the prefix will be called cutoffs, we shall mark them red.
1. Initially, the prefix contains only the root, labelled by $X$. We set $\mathcal{E} := \{X\}$.

2. Select a node $n$ on the prefix that is not the target of a cutoff edge. Let $B(n) = Y$ be the label of the node, and let $Z$ be a state with $Y \rightarrow Z$ such that the prefix does not contain any edge from $n$ to a $Z$-labelled node.

2a. If no such pair $n, Z$ exists, we are done.

2b. Otherwise, add a new, $Z$-labelled node to the prefix and add an edge from $n$ to it.

2c. If $Z \in \mathcal{E}$, then the new edge is a cutoff. Otherwise, set $\mathcal{E} := \mathcal{E} \cup \{Z\}$.

3. Continue at step 2.
Example

Step-by-step construction of the prefix in the previous example:

Observation (1): A complete prefix contains as many transitions as $\mathcal{T}$.

Observation (2): The shape of the prefix depends on the order in which edges are added!
We generalize unfoldings for Petri nets, as follows:

The unfolding of a Petri net $\mathcal{P}$ (or, a prefix of the same) is an infinite acyclic Petri net $\mathcal{U}$. We shall be interested in computing a finite prefix $\mathcal{Q}$ of $\mathcal{U}$.

Remark: In the following, we call the places of $\mathcal{Q}$ conditions, the transitions of $\mathcal{Q}$ events. This merely serves to better distinguish the elements of $\mathcal{P}$ and $\mathcal{Q}$, functionally they are the same!
Every condition of $Q$ is labelled by a place of $P$, every event of $Q$ by a transition of $P$.

Every event $e$ is of the form $(S, t)$, where $S$ is the preset of $e$ and $t$ the label of $e$.

Let $S$ be a set of conditions. $B(S)$ denotes the set of places labelling the elements of $S$.

Every condition has exactly one incoming arc.

Some events are labelled as cutoffs.
Prefix construction for Petri nets

1. Let $m_0$ be the initial marking of $\mathcal{P}$. Then $\mathcal{Q}$ initially contains one condition for each place in $m_0$. The initial marking of $\mathcal{Q}$ contains exactly these conditions. We set $\mathcal{E} := \{m_0\}$.

2. Let $t$ be a transition of $\mathcal{P}$ and $S$ a set of conditions none of which is the output place of a cutoff transition. Moreover, let $S$ be part of a reachable marking in $\mathcal{Q}$, let $B(S) = \cdot t$, and suppose that $(S, t)$ is not yet contained in $\mathcal{Q}$.

2a. If no such pair $(S, t)$ exists, we are done.

2b. Add the event $e := (S, t)$ to the prefix (with $S$ as preset and label $t$). Moreover, extend the prefix by one condition for every output place of $t$ and make it an output place of $e$.

2c. We associate with $e$ a marking $m_e$ (which is reachable in $\mathcal{P}$) (see below). If $m_e \in \mathcal{E}$, then $e$ is a cutoff. Otherwise $\mathcal{E} := \mathcal{E} \cup \{m_e\}$.
Remark: If we omit step 2c (no cutoffs), then we obtain the full unfolding $U$. The shape of $Q$ again depends on the order in which events are added. (More on this in a moment!)
Example 1: Petri net...
... and a possible prefix of the unfolding
Determining $m_e$

For the event $e = (S, t)$, we determine $m_e$, a marking of $P$, as follows:

Idea: $m_e$ is the label of the marking obtained by making the “minimal” effort to fire $e$.

Let $x, y$ be two nodes (conditions or events) in $Q$. Let $<$ be the smallest partial order where $x < y$ if there is an edge from $x$ to $y$.

Let $x$ be a node of $Q$. We define $\lfloor x \rfloor := \{ y \mid y \leq x \}$.

Let $m_e$ be the labels of the marking obtained by firing the events of $\lfloor e \rfloor$ (in any order). Note: Such a firing sequence exists due to the properties of $S$. 
Complete prefixes

Let \( \mathcal{P} \) be a Petri net and \( \mathcal{Q} \) a prefix of its unfolding \( \mathcal{U} \) with labelling \( B \). We call \( \mathcal{Q} \) complete if it satisfies the following property:

A marking \( m \) is reachable in \( \mathcal{P} \) iff a marking \( m' \) with \( B(m') = m \) is reachable in \( \mathcal{Q} \).

Thus, if \( \mathcal{Q} \) is complete, we can decide reachability in \( \mathcal{P} \) by examining \( \mathcal{Q} \); the latter problem is NP-complete rather than PSPACE-complete.

Unfortunately, the algorithm given previously does not always produce a complete prefix. Indeed, its shape depends on the order in which events are added. We shall discuss an example that demonstrates this effect.
Example 2

Consider the following Petri net:

```
c d
B
A
k
a b c d
E F
p
l
T
C
k
E
C
D
T
f
e
```
In Example 2 the marking $\{p\}$ is reachable, e.g. by firing $A B T$.

The net can also reach the marking $\{e, f\}$ by firing either $A C$ or $B D$, and then return by firing $E F$ to the initial marking.

We shall see that a prefix generated according to depth-first order will “overlook” the transition $T$. 
Depth-first order generated the prefix shown below (order and cutoffs indicated in red):
Configurations

Let \( m \) be a reachable marking in \( \mathcal{U} \) and let \( C \) be the set of events in \( \bigcup_{c \in m} [c] \). Then we call \( C \) a configuration.

A configuration \( C \) represents a set of events that can all fire in one execution. Given \( C \), we denote the marking (of \( \mathcal{U} \)) reached by such an execution by \( m_C \).

Remark: For every event \( e \), the set \( [e] =: C_e \) is a configuration. We have \( B(m_{C_e}) = m_e \).

We call \( E \) an extension of \( C \) iff \( C \cap E = \emptyset \) and \( C \cup E \) is a configuration. In this case, we write \( C \oplus E \) to denote the configuration \( C \cup E \).

Let \( C, C' \) be two configurations such that \( B(m_C) = B(m_{C'}) \). If \( E \) is an extension of \( C \), then there is an extension \( E' \) of \( C' \) that is isomorphic to \( E \).
Adequate orders

Let \( \prec \) be a well-founded total order on configurations that refines \( \subset \) (i.e. \( C \subset C' \) implies \( C \prec C' \)).

**Intuition:** \( \prec \) is a possible order in which the events of \( \mathcal{U} \) can be generated; i.e., \( e \) is added before \( e' \) if \( C_e \prec C_{e'} \).

Let \( Q \prec \) be the prefix of \( \mathcal{U} \) generated by adding events in the order given by \( \prec \) as above.

We call \( \prec \) adequate iff \( Q \prec \) is complete.
A sufficient condition for adequate orders

The following condition guarantees that $\prec$ is adequate:

Let $C, C'$ be two configurations with $B(m_C) = B(m_{C'})$, and let $E$ an extension of $C$ and $E'$ the extension of $C'$ isomorphic to $E$. Then $D \prec D'$ must hold, where $D = C \oplus E$ and $D' = C' \oplus E'$.

**Proof:** Let $\prec$ be an order satisfying the above constraint. We show that $Q_\prec$ is complete. So let $m$ be a marking reachable in $\mathcal{P}$. Then there is a marking $m'$ in $\mathcal{U}$ with $B(m') = m$. Let $C$ be the configuration built from events in $\bigcup_{c \in m'} [c]$. Either $C$ is contained in $Q_\prec$, or $C = C_e \oplus E$ for some cutoff event $e$. But then there is another event $e'$ with $m_{e'} = m_e$ and $C_{e'} \prec C_e$ and therefore a configuration $D := C_{e'} \oplus E'$, where $E'$ is isomorphic to $E$, and we have $B(m_D) = B(m_C) = m$. Either $D$ is contained in $Q_\prec$, or one repeats the argument, but only finitely often since $\prec$ is well-founded.
Conflict, causality, concurrency

From the structure of the unfolding we can derive statements about the mutual relationships of conditions:

Let \( c, d \) be two (different) conditions of \( Q \).

\( c, d \) are called **causally dependent** if \( c < d \) or \( d < c \). (I.e., in every firing sequence containing both conditions, one condition must be consumed to generate the other.)

\( c, d \) are **in conflict** if there are events \( e, f \) (where \( e \neq f \)), \( e \in \downarrow c \), \( f \in \downarrow d \), and \( \bullet e \cap \bullet f \neq \emptyset \). (I.e., \( c, d \) can never occur in a reachable marking!)

\( c, d \) are called **concurrent** if they are neither causally dependent nor in conflict with one another (I.e., \( c, d \) can occur together in some reachable marking!)
Reachability checking using complete prefixes

Theorem: Let $P$ be a Petri net and $Q$ a complete unfolding prefix. Given $Q$ and a marking $m$ of $P$, it is NP-complete to determine whether $m$ is reachable in $P$.

Proof: Membership in NP: guess a marking $m'$ of $Q$ such that $B(m') = m$, check if it does not contain causally dependent or conflicting conditions.

Hardness: polynomial reduction from SAT (proof on blackboard)
Reducing reachability to SAT

In the other direction, we can, given $m$ and $Q$, produce a propositional logic formula, of polynomial size in $|m| + |Q|$, that is satisfiable iff $m$ is reachable in $P$. This makes sense because extremely efficient SAT solvers are available (like minisat).

The formula uses one boolean variable for each event and each condition. Its satisfying assignments are those that correspond to conditions and their associated markings. Finally, one demands that the image of the marking is $m$.

Remark (1): Notice that the unfolding and most of the formula is independent of $m$ and needs to be generated from $P$ only once for an arbitrary number of reachability queries.

Remark (2): In a very similar way, one can check whether $P$ contains a deadlock, i.e. a reachable marking that does not enable any transition.
Unbounded nets: Coverability graphs
Use of reachability graphs

If the net is not $k$-safe for any $k$, then it has infinitely many reachable markings. The unfolding method will therefore not terminate, and neither can one effectively compute the reachability graph.

Nevertheless, the following problem is decidable: Given a (non-safe) net $\mathcal{P}$ and a marking $m$, is $m$ reachable in $\mathcal{P}$?

This result is due to Mayr and Kosaraju (1981/82). However, the complexity of the problem is in general non-elementary, and no efficient methods are known.

Most of the time, though, one is interested in checking whether $m$ is part of a reachable marking (one says that $m$ is coverable in this case). This problem is somewhat easier to solve in practice, and we shall discuss it.
Example

Consider the following (slightly inept) attempt at modelling a traffic light:

R → RY
RY → G
Y → R
G → Y
p3 (green light)
p2 (yellow light)
p1 (red light)
Coverability Graphs

The reachability graph of the preceding net is infinite.

We will show the construction of a so-called coverability graph for it.

The coverability graph has the following properties:

It can be used to find out whether the reachability graph is infinite.

It is always finite, and its construction always terminates.

Even for unbounded nets, it still gathers some information about reachable markings.
Computing with $\omega$

First we introduce a new symbol $\omega$ to represent “arbitrarily many” tokens.

We extend the arithmetic on natural numbers with $\omega$ as follows. For all $n \in \mathbb{N}$:

$n + \omega = \omega + n = \omega$,

$\omega + \omega = \omega$,

$\omega - n = \omega$,

$0 \cdot \omega = 0, \omega \cdot \omega = \omega$,

$n \geq 1 \Rightarrow n \cdot \omega = \omega \cdot n = \omega$,

$n \leq \omega$, and $\omega \leq \omega$.

Note: $\omega - \omega$ remains undefined, but we will not need it.
We extend the notion of markings to $\omega$-markings. In an $\omega$-marking, each place $p$ will either have $n \in \mathbb{N}$ tokens, or $\omega$ tokens (arbitrarily many).

Note: This is a technical definition that we will need for constructing the coverability graph! The nets that we use only have finite markings.

An $\omega$-marking such as $(1, \omega, 0)$ can also be interpreted as the set of (non-$\omega$)-markings that have one token on the first place, no token on the third place, and any number of tokens on the second place.
Firing Rule with $\omega$-markings

The firing condition and firing rule (reproduced below) neatly extend to $\omega$-markings with the extended arithmetic rules:

Firing condition:
Transition $t \in T$ is $M$-enabled, written $M \xrightarrow{t}$, iff $\forall p \in \cdot t : M(p) \geq W(p, t)$.

Firing rule:
An $M$-enabled transition $t$ may fire, producing the successor marking $M'$, where
\[
\forall p \in P : M'(p) = M(p) - W(p, t) + W(t, p).
\]

If a transition has a place with $\omega$ tokens in its preset, that place is considered to have sufficiently many tokens for the transition to fire, regardless of the arc weight.

If a place contains an $\omega$-marking, then firing any transition connected with an arc to that place will not change its marking.
Definition of Covering

An $\omega$-marking $M'$ covers an $\omega$-marking $M$, denoted $M \leq M'$, iff

$$\forall p \in P: M(p) \leq M'(p).$$

An $\omega$-marking $M'$ strictly covers an $\omega$-marking $M$, denoted $M < M'$, iff

$$M \leq M' \quad \text{and} \quad M' \neq M.$$
Observation: Let $M$ and $M'$ be two markings such that $M \leq M'$. Then for all transitions $t$, the following holds:

$$\text{If } M \xrightarrow{t} \text{ then } M' \xrightarrow{t}.$$ 

In other words, if $M'$ has at least as many tokens as $M$ has (on each place), then $M'$ enables at least the same transitions as $M$ does.

This observation can be extended to sequences of transitions: Define $M \xrightarrow{t_1t_2\ldots t_n} M'$ to denote:

$$\exists M_1, M_2, \ldots, M_n : M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \ldots \xrightarrow{t_n} M_n = M'.$$

Now, if $M \xrightarrow{t_1t_2\ldots t_n}$ and $M \leq M'$, then $M' \xrightarrow{t_1t_2\ldots t_n}$. 
Let $M, M'$ be markings such that $M < M'$, and assume that there is a sequence of transitions such that $M \xrightarrow{t_1 t_2 \ldots t_n} M'$ holds.

Thus, there is a marking $M''$ with $M' \xrightarrow{t_1 t_2 \ldots t_n} M''$.

Let $\Delta M := M' - M$ (place-wise difference). Because $M < M'$, the values of $\Delta M$ are non-negative and at least one value is non-zero.

Clearly, $M'' = M' + \Delta M = M + 2\Delta M$. 

\[
\begin{array}{cccccc}
M & \xrightarrow{t_1 t_2 \ldots t_n} & M' & \xrightarrow{t_1 t_2 \ldots t_n} & M'' & \xrightarrow{t_1 t_2 \ldots t_n} \cdots \\
\Delta M & | & | & \Delta M & | & | \\
M + \Delta M & \xrightarrow{t_1 t_2 \ldots t_n} & M + 2\Delta M & \xrightarrow{t_1 t_2 \ldots t_n} \cdots
\end{array}
\]
By firing the transition sequence $t_1 t_2 \ldots t_n$ repeatedly we can “pump” an arbitrary number of tokens to all the places having a non-zero marking in $\Delta M$.

The basic idea for constructing the **coverability graph** is now to replace the marking $M'$ with a marking where all the places with non-zero tokens in $\Delta M$ are replaced by $\omega$. 
Coverability Graph Algorithm (1/2)

The subroutine AddOmegas(M, M', V) will check if the sequences leading to M' can be repeated, strictly increasing the number of tokens on some places, and replace their values with ω.

Coverability-Graph(⟨P, T, F, W, M0⟩)
1. ⟨V, E, ν₀⟩ := ⟨{M₀}, ∅, M₀⟩;
2. Work : set := {M₀};
3. while Work ≠ ∅
   4. do select M from Work;
   5. Work := Work \ {M};
   6. for t ∈ enabled(M)
      7. do M' := fire(M, t);
      8. M' := AddOmegas(M, M', V);
      9. if M' ∉ V
         10. then V := V ∪ {M'}
         11. Work := Work ∪ {M'};
         12. E := E ∪ {⟨M, t, M'⟩};
13. return ⟨V, E, ν₀⟩;
Coverability Graph Algorithm (2/2)

The following notation is used in the AddOmegas subroutine:

- \( M'' \to^* M \) iff the coverability graph currently contains a path (including the empty path!) leading from \( M'' \) to \( M \).

\[
\text{AddOmegas}(M, M', V)
1\text{ repeat saved } := M';
2\text{ for all } M'' \in V \text{ s.t. } M'' \to^* M
3\text{ do if } M'' < M'
4\text{ then } M' := M' + (M' - M'') \cdot \omega;
5\text{ until saved } = M';
6\text{ return } M';
\]

In other words, repeated check all the predecessor markings of the new marking \( M' \) to see if they are strictly covered by \( M' \). Line 5 causes all places whose number of tokens in \( M' \) is strictly larger than in the “parent” \( M'' \) to contain \( \omega \).
Properties of the coverability graph (1)

Let $N = \langle P, T, F, W, M_0 \rangle$ be a net.

The coverability graph has the following fundamental property:

If a marking $M$ of $N$ is reachable, then $M$ is covered by some vertex of the coverability graph of $N$.

Note that the reverse implication does not hold: A marking that is covered by some vertex of the coverability graph is not necessarily reachable, as shown by the following example:
Properties of the coverability graph (2)

The coverability graph could thus be said to compute an overapproximation of the reachable markings.

The construction of the coverability graph always terminates.
If $N$ is bounded, then the coverability graph is identical to the reachability graph.

Coverability graphs are not unique, i.e. for a given net there may be more than one coverability graph, depending on the order of the worklist and the order in which firing transitions are considered.