Unfolding based model checking

Javier Esparza

Faculty of Computer Science
Technical University of Munich
esparza@in.tum.de

Joint work with: Keijo Heljanko
Aalto University, School of Science
Keijo.Heljanko@tkk.fi

December 4, 2012
Tutorial material

- Tutorial mainly based on the book

  Esparza, J. and Heljanko, K.: Unfoldings –
  A Partial-Order Approach to Model Checking.
  EATCS Monographs in Theoretical Computer Science,

- Final book draft available from:
  http://www.model.in.tum.de/~esparza/bookunf.html

- Consult the book for bibliographical and historical information
  about the unfolding method
The state explosion problem

- A concurrent system with $N$ sequential components, each of them with $K$ states, may have up to $K^N$ reachable states.
- Hinders verification by construction of the state space even for relatively small systems.
- Approaches to fight state explosion:
  - **Abstraction**: Aggregate “similar” states.
  - **Reduction**: Remove “irrelevant” states.
  - **Compression**: Find “compact” representations of the state space.

Abstraction and reduction lose information (on purpose), compression does not.
Compression techniques

- Binary Decision Diagrams. Exploit regularity.
  - Identical components.
  - Simple communication topology: array, ring, …

- Unfoldings: Exploit concurrency.
  - Loosely coupled but possibly heterogeneous components.
The unfolding method

**Sequential systems**

Model: transition systems

Semantics: computation tree (unfolding of the TS)

Algorithmic principle: search in trees

**Concurrent systems**

Model: products of transition systems (represented as Petri nets)

Semantics: (concurrent) unfolding

Algorithmic principle: search in unfoldings
A transition system is a tuple $\mathcal{A} = \langle S, T, \alpha, \beta, is \rangle$, where

- $S$ is a set of states,
- $T$ is a set of transitions,
- $\alpha : T \rightarrow S$ associates to each transition its source state,
- $\beta : T \rightarrow S$ associates to each transition its target state, and
- $is \in S$ is the initial state
Example

Transition system $\mathcal{A} = \langle S, T, \alpha, \beta, is \rangle$ where

- $S = \{s_1, s_2, s_3, s_4\}$, $T = \{t_1, t_2, t_3, t_4, t_5\}$,
- $\alpha(t_1) = s_1$, $\beta(t_1) = s_2$, $\beta(t_5) = s_1$,
- $is = s_1$
Products of transition systems

A product of transition systems is a tuple $\langle A_1, \ldots, A_n, T \rangle$ where

- $A_1, \ldots, A_n$ are transition systems called components, and
- $T$ is a synchronization constraint.

A synchronization constraint is a set of tuples of the form $\langle u_1, u_2, \ldots, u_n \rangle$ where $u_i$ is either

- a transition of $A_i$, or
- the special idling symbol $\epsilon$.

Example: $\langle t_1, \epsilon, \epsilon, t_2 \rangle$

The tuples of $T$ are called global transitions.

A tuple $\langle s_1, s_2, \ldots, s_n \rangle$ of local states is called a global state.
Running example

\[ T = \{ \langle t_1, \epsilon \rangle, \langle t_2, \epsilon \rangle, \langle t_3, u_2 \rangle, \langle t_4, u_2 \rangle, \langle t_5, \epsilon \rangle, \langle \epsilon, u_1 \rangle, \langle \epsilon, u_3 \rangle \} \]
Peterson's mutex algorithm

```
PETerson's mutex algorithm

REQ0
b0:=T
b0:=F
t:=0
ENT0
b1=F
t=1

REQ1
b1:=T
b1:=F
t:=0
ENT1
b0=F
t=1

b0:=T

b0:=F

b1:=F

b1:=T

b0:=F

b1:=F

b0=T

b1=T

b1=F

b1=F

b1=T

b1=T
```
Petri net representation of products

Global transition $\rightarrow$ Petri net transition
Initial global state $\rightarrow$ Initial marking
Reachable global state $\rightarrow$ Reachable marking
Unfolding transition systems: Computation tree
Unfolding as a Petri net
Unfolding products
Unfolding products

\[ \langle t_1, \epsilon \rangle \langle t_2, \epsilon \rangle \langle t_3, u_2 \rangle \langle t_4, u_2 \rangle \langle t_5, \epsilon \rangle \]

\[ s_1 \]

\[ s_2 \]

\[ s_3 \]

\[ s_4 \]

\[ r_1 \]

\[ r_2 \]

\[ r_3 \]

\[ s_1 \]

\[ s_2 \]

\[ r_1 \]

\[ s_1 \]

\[ s_2 \]
Unfolding products
Unfolding products
Unfolding products

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_1, \epsilon \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_1, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]

\[ \langle t_2, \epsilon \rangle \]

\[ \langle t_3, u_2 \rangle \]

\[ \langle t_4, u_2 \rangle \]

\[ \langle t_5, \epsilon \rangle \]
Unfolding products
Unfolding products

\[
\langle t_3, u_2 \rangle \quad s_4 \quad r_3
\]

\[
\langle t_4, u_2 \rangle \quad s_4 \quad r_3
\]

\[
\langle t_2, \epsilon \rangle \quad s_1 \quad r_1
\]

\[
\langle t_1, \epsilon \rangle \quad s_1 \quad r_1
\]

\[
\langle t_5, \epsilon \rangle \quad s_2 \quad r_2
\]

\[
\langle t_1, \epsilon \rangle \quad s_2 \quad r_2
\]
The Unfolding

Transitions of the unfolding are called events.
The Unfolding

Reachable markings of the unfolding are labeled with global states of the product.
Product, net and unfolding are beh. equivalent for all the usual equivalence notions.
Unfoldings are synchronizations of trees
Causality, conflict, and concurrency

Let $x$ and $y$ be two nodes of an unfolding.

- $x$ is a causal predecessor of $y$, denoted by $x < y$, if there is a (non-empty) path from $x$ to $y$.
- $x$ and $y$ are in conflict, denoted by $x \# y$, if there are proper paths from some place $z$ to $x$ and $y$ that exit $z$ by different arcs.
- $x$ and $y$ are concurrent, denoted by $x \text{ co } y$, if neither $x \leq y$ nor $x \geq y$ nor $x \text{ co } y$.

Proposition

A set of places of an unfolding can be simultaneously marked if and only if its elements are pairwise concurrent.
Causality, conflict, and concurrency
A set of events is a configuration if
- it is causally closed (if $e \in C$ and $e' < e$ then $e' \in C$), and
- conflict-free (no two events of $C$ are in conflict)

**Proposition**

A set of events of an unfolding can be fired if and only if it is a configuration.

**Observe**: the “past” of an event (its causal predecessors) is a configuration.

We call it the local configuration of $e$. 
Configurations

Examples:

\{ 1, 4, 6, 7 \}
\{ 2, 3, 5 \}

Counterex.:

\{ 4 \}
\{ 1, 2, 3, 4 \}
\{ 1, 4, 6 \}
Checking properties
Model checking

The model checking problem:

Does some run of the system satisfy a given property?

The problem for a large class of properties can be reduced to:

1. **Executability**: Does some run contain a given transition?
2. **Repeated executability**: Does some run contain a given transition infinitely often?
3. **Livelock**: Does some run contain an infinite tail of “silent” transitions?

**Fact:**
The model-checking problem for next-free LTL-formulas can be reduced to (2) and (3), for safety properties to (1).
Program for the rest of the tutorial

Unfolding-based algorithms for

- Executability (long)
  - Search procedures
  - Adequate strategies
- Repeated executability (1 slide)
- Model checking (2 slides)

More on checking safety properties:

- Designing unfolders
- Compressing the state space: canonical prefixes
- Deciding properties with canonical prefixes
Executability
Executability in transition systems
Executability in transition systems
Executability in transition systems
Executability in transition systems
Executability in transition systems
Executability in transition systems
Executability in transition systems
Search procedures

The executability problem for transition systems can be solved by depth-first-search (DFS), breadth-first-search (BFS), or some other search procedure.

Conducting a DFS or BFS amounts to exploring a prefix of the computation tree.

The executability problem for products can also be solved by search procedures that explore a prefix of the Unfolding.

We need a formalization of search procedure.
Search procedures

A search procedure consists of:

(1) a search scheme

- **Termination condition**: Determines which leaves of the current prefix are terminals, i.e., nodes whose successors need not be explored. (Terminals are also called cut-offs.)
- **Success condition**: Determines which terminals are successful, i.e., terminals proving that $\psi$ holds.

(2) a search strategy

- determines which possible extension of the current prefix is added to it. (nondeterministic search strategies allowed!).
Search procedure for executability in transition systems

Search procedure to decide if some run executes a goal transition $g$.

Search scheme:
An event is a terminal if

1. it is labeled by $g$ or,
2. it leads to the same state as some other event already explored

A terminal is successful if it is of type (1).

Search strategy: Any.
Example (again)
Example (again)
Example (again)
Example (again)
Example (again)
Example (again)
Example (again)
Second example with $g = \{ t_5 \}$
Second example: Two prefixes

(a)

(b)
Search procedure for executability in transition systems

Search procedure to decide if some run executes a goal transition $g$.

Search scheme:

An event is a terminal if

(1) it is labeled by $g$ or,

(2) it leads to the same state as some other event already explored

A terminal is successful if it is of type (1).

Search strategy: Any.

Easy to show: All these search procedures (different strategies, same scheme) are correct (terminate with the right outcome, but may explore different sets of nodes).
Generalization to products: search scheme

We want something like this:

An event is a terminal if

(1) it is labeled by \( g \) (and then it is successful) or,

(2) it leads to the same global state (marking) as some other event already explored.

A terminal is successful if it is of type (1).

But what does it mean “it leads to the same marking as some other event already explored”?

An event does not always leads to only one marking!
Solution: attach to an event the global state reached by “executing its past”. (McMillan ’92,’95)

This is the global state reached by firing the local configuration of the event.
Solution: attach to an event the global state reached by “executing its past”. (McMillan ’92,’95)

This is the global state reached by firing the local configuration of the event.
\[
\langle t_1, \epsilon \rangle \xrightarrow{s_1} \langle t_2, \epsilon \rangle \xrightarrow{r_1} \langle \epsilon, u_1 \rangle
\]

\[
\langle t_3, u_2 \rangle \xrightarrow{r_2} \langle t_4, u_2 \rangle
\]

\[
\langle s_2, r_1 \rangle \xrightarrow{s_2} \langle s_3, r_1 \rangle \xrightarrow{s_3} \langle s_1, r_2 \rangle
\]

\[
\langle s_4, r_3 \rangle
\]

\[
\langle s_4, r_3 \rangle
\]

\[
\langle s_4, r_3 \rangle
\]

\[
\langle s_4, r_3 \rangle
\]
Generalization to products: search strategies

A search strategy determines which possible extension is added to the current prefix.

Mathematical definition?
Generalization to products: search strategies

A search strategy determines which possible extension is added to the current prefix.

Mathematical definition?

Transition systems: an event is characterized by its past, the unique transition sequence leading to it.

Search strategy: (partial) order \( \prec \) on transition sequences that refines the prefix order.
Two search strategies for $w, w' \in T^*$:

- $w \prec w'$ if $|w| < |w'|$
- $w \prec w'$ if $w$ is lexicographically smaller than $w'$
Generalization to products: search strategies

A search strategy determines which possible extension is added to the current prefix.

Mathematical definition?

Transition systems: an event is characterized by its past, the unique transition sequence leading to it.

Search strategy: (partial) order \( \prec \) on transition sequences that refines the prefix order.
**Generalization to products: search strategies**

A search strategy determines which possible extension is added to the current prefix.

**Mathematical definition?**

**Transition systems**: an event is characterized by its past, the unique transition sequence leading to it.

---

**Search strategy**: (partial) order $\prec$ on transition sequences that refines the prefix order.

---

**Products**: an event is also characterized by its past, but the past may consist of many transition sequences!
The past of event labelled $\langle t_3, u_2 \rangle$ are the transition sequences:

- $w_1 = \langle t_1, \epsilon \rangle \langle \epsilon, u_1 \rangle \langle t_3, u_2 \rangle$
- $w_2 = \langle \epsilon, u_1 \rangle \langle t_1, \epsilon \rangle \langle t_3, u_2 \rangle$
Generalization to products: search strategies

A search strategy determines which possible extension is added to the current prefix.

Mathematical definition?

**Transition systems:** an event is characterized by its past, the unique transition sequence leading to it.

Search strategy: (partial) order \( \prec \) on transition sequences that refines the prefix order.

**Products:** an event is also characterized by its past, but the past may consist of many transition sequences!
Generalization to products: search strategies

A search strategy determines which event out of the possible extensions is added to the current prefix.

Mathematical definition?

**Transition systems**: an event is characterized by its past, the unique transition sequence leading to it.

Search strategy: (partial) order $\prec$ on transition sequences that refines the prefix order.

**Products**: an event is also characterized by its past, but the past consists of many transition sequences!

**Solution**: these sequences build a Mazurkiewicz trace.

Search strategy: (partial) order $\prec$ on Mazurkiewicz traces that refines the prefix order.
Mazurkiewicz traces

- Two global transitions of a product are independent if no component participates in both of them.
  - Example: $\langle t_1, \epsilon \rangle$ and $\langle \epsilon, u_1 \rangle$ are independent, $\langle t_1, \epsilon \rangle$ and $\langle t_3, u_2 \rangle$ are not.

- Two sequences of global transitions are equivalent if the one can be obtained from the other by repeatedly swapping adjacent independent transitions.
  - Example: $\langle t_1, \epsilon \rangle \langle \epsilon, u_1 \rangle \langle t_3, u_2 \rangle \sim \langle \epsilon, u_1 \rangle \langle t_1, \epsilon \rangle \langle t_3, u_2 \rangle$

- Mazurkiewicz trace: equivalence class of sequences.
  - Example: $[\langle t_1, \epsilon \rangle \langle \epsilon, u_1 \rangle \langle t_3, u_2 \rangle] = \left\{ \langle t_1, \epsilon \rangle \langle \epsilon, u_1 \rangle \langle t_3, u_2 \rangle, \langle \epsilon, u_1 \rangle \langle t_1, \epsilon \rangle \langle t_3, u_2 \rangle \right\}$
Search strategy:

\[ [w] \prec [w'] \Leftrightarrow |w| < |w'| \]

(Well defined because equivalent sequences have the same length)
Are these search procedures correct?

Not for every strategy!!
\[
T = \{ a = \langle a_1, a_2, a_3, a_4 \rangle, b = \langle b_1, b_2, b_3, b_4 \rangle, c = \langle c_1, c_2, \epsilon, \epsilon \rangle, \\
d = \langle \epsilon, \epsilon, d_3, d_4 \rangle, e = \langle e_1, e_2, \epsilon, \epsilon \rangle, f = \langle \epsilon, \epsilon, f_3, f_4 \rangle, \\
g = \langle g_1, \epsilon, g_3, \epsilon \rangle, h = \langle \epsilon, h_2, \epsilon, h_4 \rangle, i = \langle i_1, i_2, i_3, i_4 \rangle \} \\
G = \{ i \}
\]
Which are the correct strategies?

Sufficient condition: adequate strategies

Mazurkiewicz traces can be concatenated in the obvious way:

\[ [w][w'] \overset{\text{def}}{=} [ww'] \]

A strategy \( \prec \) on Mazurkiewicz traces is adequate if it is

1. well-founded
   (no infinite descending chain \([w_0] \succ [w_1] \succ [w_2] \succ \cdots\))
2. preserved by extensions
   (\([w'] \prec [w] \) implies \([w'][w''] \prec [w][w'']\) for every \([w'']\)).
Which are the correct strategies?

Sufficient condition: adequate strategies

Mazurkiewicz traces can be concatenated in the obvious way:

\[ [w][w'] \overset{\text{def}}{=} [w w'] \]

A strategy \( \prec \) on Mazurkiewicz traces is adequate if it is

1. **well-founded**
   (no infinite descending chain \([w_0] \succ [w_1] \succ [w_2] \succ \cdots\))

2. **preserved by extensions**
   ([\(w'\)] \prec [w] implies \([w'][w''] \prec [w][w'']\) for every \([w'']\)).

(Lemma [Chatain and Khomenko]: (1) \(\rightarrow\) (2).)

Theorem: The search procedure is correct for every adequate strategy.

Proof idea:
To prove: if $g$ can be executed, then the search procedure explores some trace $[u \, g]$. 
**Theorem:** The search procedure is correct for every adequate strategy.

**Proof idea:**

To prove: if $g$ can be executed, then the search procedure explores some trace $[u g]$.

If $g$ can be executed, then the **Unfolding** has some trace $[w g]$. 
**Theorem:** The search procedure is correct for every adequate strategy.

**Proof idea:**

To prove: if $g$ can be executed, then the search procedure explores some trace $[u g]$.

If $g$ can be executed, then the Unfolding has some trace $[w g]$.

If $[w g]$ is explored, we are done. Otherwise, $w$ contains a terminal event. Let $[w_1]$ be its past. There exists another trace $[w'_1] \prec [w_1]$ such that:

- $[w g] = [w_1 \ w_2 g]$,
- $[w'_1]$ leads to the same global state as $[w_1]$. 
Theorem: The search procedure is correct for every adequate strategy.

Proof idea:

To prove: if $g$ can be executed, then the search procedure explores some trace $[u \ g]$.

If $g$ can be executed, then the Unfolding has some trace $[w \ g]$.

If $[w \ g]$ is explored, we are done. Otherwise, $w$ contains a terminal event. Let $[w_1]$ be its past. There exists another trace $[w'_1] \prec [w_1]$ such that:

- $[w \ g] = [w_1 \ w_2 \ g]$,
- $[w'_1]$ leads to the same global state as $[w_1]$.

Since $\prec$ is preserved by extensions, $[w'_1 \ w_2 \ g] \prec [w_1 \ w_2 \ g]$.

Iterating the procedure, and by well-foundedness of $\prec$, we eventually reach some trace $[u \ g]$ that is explored.
Search procedure for executability in products

Search procedure to decide if some run executes a goal transition $g$.

Search strategy: Any adequate strategy $\prec$.

Search scheme: An event $e$ is a terminal if
1. it is labeled by $g$ or,
2. some event $e' \prec e$ satisfies $\text{St}(e') = \text{St}(e)$.

A terminal is successful if it is of type (1).
An adequate strategy

Consider again the strategy:

- $[w] < [w'] \iff |w| < |w'|$

  (well defined because equivalent sequences have the same length)

An event is a terminal if some strictly smaller event with the same marking has already been explored.

But then the prefix may be exponentially larger than the number of reachable markings!!
A prefix can be very large . . .
Total adequate strategies

The problem of the previous strategy is that the adequate order is not total.
Total adequate strategies

The problem of the previous strategy is that the adequate order is not total.

An event is a terminal if some strictly smaller event with the same marking has already been explored.
Total adequate strategies

The problem of the previous strategy is that the adequate order is not total.

An event is a terminal if some strictly smaller event with the same marking has already been explored.

⇒ If the order is total, no two events of the prefix have the same marking.
The problem of the previous strategy is that the adequate order is not total.

An event is a terminal if some strictly smaller event with the same marking has already been explored.

⇒ If the order is total, no two events of the prefix have the same marking.

⇒⇒ The prefix can contain at most as many events as the number of reachable markings.
Are there total adequate strategies?
Fact 1: Every total adequate strategy on transition sequences can be lifted to a total adequate strategy on Mazurkiewicz traces:

- Given $w$, consider its projections $w_1, w_2, \ldots, w_n$ on the components of the product. Example:

- $[w'] \prec [w]$ if there is an index $i$ such that

  $$w_1' = w_1, w_2' = w_2, \ldots, w_{i-1}' = w_{i-1} \quad \text{and} \quad w_i' \prec w_i$$
Are there total adequate strategies?

Fact 2: The following strategy is adequate and total on transition sequences: $w_1 \prec w_2$ iff

- $|w_1| < |w_2|$, or
- $|w_1| = |w_2|$ and $w_1$ is lexicographically smaller than $w_2$. 
There are many total adequate strategies!

Esparza, Römer, and Vogler: Based on Foata normal forms.

Esparza, Römer: Distributed strategies.

Niebert, Qu: $[w_1] \prec [w_2]$ iff

- the Parikh vector of $[w_1]$ is lexicographically smaller than the Parikh vector of $[w_2]$, or
- the Parikh vectors of $[w_1]$ and $[w_2]$ are equal, and the lexicographic smallest sequence in $[w_1]$ is lexicographically smaller than the lexicographic smallest sequence in $[w_2]$. 
Repeated executability
A search procedure for repeated executability

This procedure has a “BFS” emptiness checker flavor to it, the livelock problem has a similar algorithmic solution: Given an event $e$, let $\#_g e$ be the number of occurrences of $g$ in the past of $e$.

Search strategy: any adequate strategy.

Search scheme: An event $e$ is a terminal if there is $e' \prec e$ such that $\text{St}(e) = \text{St}(e')$ and either

1. $e' < e$, or
2. $e \not< e$, and $\#_g e' \geq \#_g e$.

A terminal is successful if it is of type (1) and some event between $e'$ and $e$ is labelled by $g$. 
Example: repeated executability of $\langle t_1, \epsilon \rangle$
Designing unfolders
Computing possible extensions

- Core of any unfoldor.
- Takes 90%+ of the running time.
- Complexity of adding one event? Algorithms?
Computing possible extensions is NP-complete

A decision version of computing the possible extensions is NP-complete in the size of the prefix. Consider the 3SAT formula

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \]
Computing possible extensions is NP-complete

A partial prefix of the system. Now $t$ is in the possible extensions iff $\phi$ is satisfiable.
Computing possible extensions

Let $k$ be the maximum in-degree of transitions and $n$ be the number of places in the prefix before calling the possible extensions subroutine.

- **Memory-intensive approach**: Maintain the co-relation between any two conditions. Takes $O(n^2)$ memory and takes $O(n^k/k^{k-2})$ time. Also updating the co-relation takes $O(n)$ time for each added condition.

- **Memory-light approach**: Enumerate all possible extensions without any co-relation using $O(n)$ memory but $O(n^{k+1}/k^k)$ time.

- **More refined search approach**: Preset trees (Khomenko)

- **Solver approach**: Employ an NP solver to compute the potential extensions.
Compressing the state space: canonical prefixes
Canonical prefixes

- Executability: If the goal transition cannot occur, the algorithm always generates the same prefix of the unfolding, even if the strategy is nondeterministic.

- This prefix “contains” all reachable global states (for every reachable global state $s$ there is a reachable marking $M$ of the prefix labeled by $s$).

- This unique prefix is called the canonical prefix (theory by Khomenko, Koutny, and Vogler).

- The ratio \[ \frac{\text{size of the canonical prefix}}{\text{number of reachable states}} \]
  measures the “degree of compression” achieved.

- Moreover: once computed, the canonical prefix can be reused to solve reachability questions, deadlock freedom, and other safety properties.
A canonical finite prefix can be very succinct

The class of Petri nets containing the following representative for $n = 4$ has a state space of size $2^n$ but a prefix of linear size in the parameter $n$:

The prefix is identical to the original net system!
Canonical finite prefix sizes

Prefixes are often smaller than the state space.
For total search strategies prefixes have never more events than reachable states.

<table>
<thead>
<tr>
<th>Problem(size)</th>
<th>S</th>
<th>T</th>
<th>B</th>
<th>E</th>
<th>#c</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPD(5)</td>
<td>45</td>
<td>45</td>
<td>1582</td>
<td>790</td>
<td>211</td>
<td>3488</td>
</tr>
<tr>
<td>DPD(6)</td>
<td>54</td>
<td>54</td>
<td>3786</td>
<td>1892</td>
<td>499</td>
<td>19860</td>
</tr>
<tr>
<td>DPD(7)</td>
<td>63</td>
<td>63</td>
<td>8630</td>
<td>4314</td>
<td>1129</td>
<td>109964</td>
</tr>
<tr>
<td>DPH(5)</td>
<td>48</td>
<td>67</td>
<td>2712</td>
<td>1351</td>
<td>547</td>
<td>3112</td>
</tr>
<tr>
<td>DPH(6)</td>
<td>57</td>
<td>92</td>
<td>14590</td>
<td>7289</td>
<td>3407</td>
<td>16896</td>
</tr>
<tr>
<td>DPH(7)</td>
<td>66</td>
<td>121</td>
<td>74558</td>
<td>37272</td>
<td>19207</td>
<td>79926</td>
</tr>
<tr>
<td>RING(7)</td>
<td>91</td>
<td>77</td>
<td>813</td>
<td>403</td>
<td>79</td>
<td>16999</td>
</tr>
<tr>
<td>RING(9)</td>
<td>117</td>
<td>99</td>
<td>1599</td>
<td>795</td>
<td>137</td>
<td>211527</td>
</tr>
<tr>
<td>ELEVATOR(2)</td>
<td>146</td>
<td>299</td>
<td>1562</td>
<td>827</td>
<td>331</td>
<td>1061</td>
</tr>
<tr>
<td>ELEVATOR(3)</td>
<td>327</td>
<td>783</td>
<td>7398</td>
<td>3895</td>
<td>1629</td>
<td>7120</td>
</tr>
<tr>
<td>ELEVATOR(4)</td>
<td>736</td>
<td>1939</td>
<td>32354</td>
<td>16935</td>
<td>7337</td>
<td>43439</td>
</tr>
<tr>
<td>FURNACE(1)</td>
<td>27</td>
<td>37</td>
<td>535</td>
<td>326</td>
<td>189</td>
<td>343</td>
</tr>
<tr>
<td>FURNACE(2)</td>
<td>40</td>
<td>65</td>
<td>4573</td>
<td>2767</td>
<td>1750</td>
<td>3777</td>
</tr>
<tr>
<td>FURNACE(3)</td>
<td>53</td>
<td>99</td>
<td>30820</td>
<td>18563</td>
<td>12207</td>
<td>30860</td>
</tr>
</tbody>
</table>
But, shouldn’t you compare with the size of a BDD?
Heterogeneous philosophers: BDD size

- 100 random tables with right-handed, left-handed, and ambidextrous philosophers
- BDD for the set of reachable states

<table>
<thead>
<tr>
<th>Nr. of phil.</th>
<th>BDD size</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>178</td>
<td>94</td>
<td>355</td>
<td>52</td>
<td>0.30</td>
</tr>
<tr>
<td>6</td>
<td>583</td>
<td>248</td>
<td>1716</td>
<td>305</td>
<td>0.52</td>
</tr>
<tr>
<td>8</td>
<td>1553</td>
<td>390</td>
<td>8678</td>
<td>1437</td>
<td>0.92</td>
</tr>
<tr>
<td>10</td>
<td>3140</td>
<td>510</td>
<td>27516</td>
<td>4637</td>
<td>1.48</td>
</tr>
<tr>
<td>12</td>
<td>4855</td>
<td>632</td>
<td>47039</td>
<td>8538</td>
<td>1.76</td>
</tr>
<tr>
<td>14</td>
<td>33742</td>
<td>797</td>
<td>429903</td>
<td>85798</td>
<td>2.54</td>
</tr>
</tbody>
</table>
Heterogeneous philosophers: Prefix size

- 100 random tables with right-handed, left-handed, and ambidextrous philosophers
- Nodes of the canonical prefix

<table>
<thead>
<tr>
<th>Nr. of phil.</th>
<th>Prefix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>6</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>95</td>
</tr>
<tr>
<td>10</td>
<td>117</td>
</tr>
<tr>
<td>12</td>
<td>141</td>
</tr>
<tr>
<td>14</td>
<td>161</td>
</tr>
</tbody>
</table>
Checking deadlock-freedom with BDDs

- 100 random tables with right-handed, left-handed, and ambidextrous philosophers
- SMV on a very old machine ...

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.08</td>
<td>0.05</td>
<td>0.13</td>
<td>0.02</td>
<td>0.29</td>
</tr>
<tr>
<td>6</td>
<td>0.36</td>
<td>0.20</td>
<td>1.18</td>
<td>0.16</td>
<td>0.46</td>
</tr>
<tr>
<td>8</td>
<td>4.14</td>
<td>1.25</td>
<td>14.60</td>
<td>2.45</td>
<td>0.59</td>
</tr>
<tr>
<td>10</td>
<td>56.60</td>
<td>15.80</td>
<td>388.00</td>
<td>46.90</td>
<td>0.83</td>
</tr>
<tr>
<td>12</td>
<td>1595.00</td>
<td>228.00</td>
<td>10616.00</td>
<td>1615.00</td>
<td>1.01</td>
</tr>
</tbody>
</table>
Checking deadlock-freedom with unfoldings

- 100 random tables with right-handed, left-handed, and ambidextrous-philosophers
- PEP on a very old machine ...

<table>
<thead>
<tr>
<th>Nr. of phil.</th>
<th>Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
</tr>
<tr>
<td>8</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
</tr>
<tr>
<td>12</td>
<td>0.02</td>
</tr>
<tr>
<td>14</td>
<td>0.02</td>
</tr>
<tr>
<td>16</td>
<td>0.02</td>
</tr>
<tr>
<td>18</td>
<td>0.03</td>
</tr>
</tbody>
</table>
External benchmarks ("real world" benchmarks)

- Analysis of asynchronous circuits (Khomenko, McMillan, Semenov, Yakovlev, and others).
- Automated testing of multithreaded programs (Kähkönen, Saarikivi, Heljanko).
- Planning (Hickmott, Rintanen, Thiébaux, White).
- Analysis of biological networks (Karlebach, Shamir).
- Fault detection in telecommunication networks (Jard and others).
- Analysis of manufacturing supply chain networks (Dong, Chen).
Deciding reachability with canonical prefixes
## Reachability

### Reachability of local states

<table>
<thead>
<tr>
<th>Product/1-safe PN</th>
<th>Canonical prefix</th>
<th>Interleaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPACE-complete</td>
<td><strong>Linear</strong></td>
<td>Linear</td>
</tr>
</tbody>
</table>

### Reachability of global states

<table>
<thead>
<tr>
<th>Product/1-safe PN</th>
<th>Canonical prefix</th>
<th>Interleaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPACE-complete</td>
<td><strong>NP-complete</strong></td>
<td>Linear</td>
</tr>
</tbody>
</table>
A canonical prefix
Reducing reachability to SAT

\[
\begin{array}{c|c}
p & \phi_p \\
\hline
\alpha & \alpha \leftrightarrow \neg e1 \\
\beta & \beta \leftrightarrow \neg e1 \\
\gamma & ((e3 \lor e4) \rightarrow e1) \land \neg (e3 \land e4) \land \\
& (\gamma \leftrightarrow (e1 \land \neg e3 \land \neg e4)) \\
\delta & ((e2 \lor e6) \rightarrow e1) \land \neg (e2 \land e6) \land \\
& (\delta \leftrightarrow (e1 \land \neg e2 \land \neg e6)) \\
\epsilon & \epsilon \leftrightarrow e2 \\
\zeta & \zeta \leftrightarrow e3 \\
\eta & (e6 \rightarrow e4) \land (\eta \leftrightarrow (e4 \land \neg e6)) \\
\kappa & ((e8 \lor e9) \rightarrow e6) \land \neg (e8 \land e9) \land \\
& (\kappa \leftrightarrow (e6 \land \neg e8 \land \neg e9)) \\
\lambda & \lambda \leftrightarrow e6 \\
\mu & \mu \leftrightarrow e8 \\
\nu & \nu \leftrightarrow e9 \\
\end{array}
\]
A conjunction of all the formulas for the conditions gives a formula encoding all reachable configurations of the prefix.

It is easy to project this on the markings of the original net by introducing variables for the original places of the net and adding to the formula a conjunction for each place of the original net:

\[
\begin{align*}
  s_1 & \iff (\alpha \lor \zeta \lor \mu) \\
  \vdots & \vdots \\
  r_2 & \iff \delta
\end{align*}
\]

A global state marking both \( s_1 \) and \( r_2 \) can be reached if the formula obtained by conjunction with \( (s_1 \land r_2) \) is satisfiable.

Deadlock detection is just another reachability property.
Deadlock checking running time

Unfolding much slower than deadlock detection (old results but the trend is still the same). Fastest tools currently are PUng (unfolding) and CLP (reachability) by Victor Khomenko

<table>
<thead>
<tr>
<th>Problem(size)</th>
<th>DL</th>
<th>UnfERVunfold</th>
<th>DCmcsmodels -n</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPD(5)</td>
<td>N</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>DPD(6)</td>
<td>N</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>DPD(7)</td>
<td>N</td>
<td>2.2</td>
<td>0.8</td>
</tr>
<tr>
<td>DPH(5)</td>
<td>N</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>DPH(6)</td>
<td>N</td>
<td>4.1</td>
<td>1.3</td>
</tr>
<tr>
<td>DPH(7)</td>
<td>N</td>
<td>101.7</td>
<td>11.3</td>
</tr>
<tr>
<td>ELEVATOR(2)</td>
<td>Y</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>ELEVATOR(3)</td>
<td>Y</td>
<td>1.3</td>
<td>0.2</td>
</tr>
<tr>
<td>ELEVATOR(4)</td>
<td>Y</td>
<td>27.4</td>
<td>1.0</td>
</tr>
<tr>
<td>FURNACE(1)</td>
<td>N</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>FURNACE(2)</td>
<td>N</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>FURNACE(3)</td>
<td>N</td>
<td>14.3</td>
<td>1.1</td>
</tr>
<tr>
<td>RING(7)</td>
<td>N</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>RING(9)</td>
<td>N</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>RW(9)</td>
<td>N</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>RW(12)</td>
<td>N</td>
<td>25.3</td>
<td>2.2</td>
</tr>
</tbody>
</table>
We have introduced unfoldings, a symbolic method to compactly represent the state space of the system using:

- Abstract system model: synchronization of transition systems
- Unfolding theory built on top of the theory of Mazurkiewicz traces
- We show the algorithmic details of unfolding procedures, and reachability checking based on SAT solvers