

Model checking using unwindings?

Igor Walukiewicz
LaBRI, Bordeaux University

- The case of interleaving semantics.
- Distributed systems and their noninterleaving semantics.
- Unwinding method for deadlock checking
- Questions about extending this to trace temporal logics.

- Given a transition system \mathcal{M} and a linear time logic formula α we want to decide if all the paths of \mathcal{M} starting in the initial state satisfy α .
- We construct a Büchi automaton \mathcal{A} for $\neg\alpha$.
- We make the product $\mathcal{M} \times \mathcal{A}$.
- We do a DFS search in the product to find a “green loop”.
- Size of \mathcal{M} is $|S_1| \cdot |S_2| \cdots |S_n|$
- Size of \mathcal{A} is $\mathcal{O}(2^{|\alpha|})$

- A process is a Kripke structure $\langle \Sigma, S, s^0, tr \subseteq S \times \Sigma \times S \rangle$.
- A distributed system of n processes is an n -tuple of processes

$$\mathcal{M} = \{ \langle \Sigma_i, S_i, s_i^0, tr_i \rangle_{i=1, \dots, n} \}$$

- A global transition relation

$$Tr_{\mathcal{M}} = \{ ((s_{i_1}, \dots, s_{i_k}), a, (s'_{i_1}, \dots, s'_{i_k})) : \\ p(a) = \{i_1, \dots, i_k\} \text{ and } s_{i_j} \xrightarrow{a} s'_{i_j} \in tr_{i_j} \}$$

- An unwinding method will give a representation of all global states in the system.

- The semantics of a transition system is its unwinding to a tree.
- One execution of a distributed system is a trace.
- The semantics of a distributed system is something like a tree of traces.
- We call it here **branching system**. It is an occurrence net, or in other words, event structure with some more decoration.

- In a PN we have places and transitions.
- In a PN we can define
 - \preceq : the flow relation
 - $\#$: the conflict relation
 - co : the concurrency relation
- An **occurrence net** is a PN such that :
 - each place has at most one predecessor
 - no place or transition is in conflict with itself (no $y\#y$)
 - the past of each place is finite.

- Fix a distributed system $\mathcal{M} = \{\langle \Sigma_i, S_i, s_i^0, tr_i \rangle_{i=1, \dots, n}\}$.
- We define an occurrence net $\llbracket \mathcal{M} \rrbracket = \langle P, E \rangle$
- Places of $\llbracket \mathcal{M} \rrbracket$ will have the form (s, e) for s a local state of \mathcal{M} and e an event.
- Events of $\llbracket \mathcal{M} \rrbracket$ will have the form (t, X) for $t \in T$ and $X \subseteq P$ such that $\text{last}(X) = \bullet t$.
- $(s_i^0, \perp) \in P$ for every $i = 1, \dots, n$.
- if $X \subseteq P$ and $t \in T$ with $\text{last}(X) = \bullet t$ then
$$(t, X) \in E$$
$$(s, (t, X)) \in P \text{ for } s \in t^\bullet.$$

Properties of branching system

Fact: $[[\mathcal{M}]]$ is an occurrence net.

Fact: Two nodes of process i are either causally related or in conflict.

• There is a notion of homomorphism $h : [[\mathcal{M}]] \rightarrow \mathcal{M}$. It sends places to local states and events to global transitions in an obvious way.

• More generally one can define a notion of a homomorphism of Petri Nets.

Fact: Branching system is the maximal occurrence net with a homomorphism into \mathcal{M} .

Cor: Branching system is to the system as tree to the Kripke structure.

- A **configuration** is a set of events C such that:
 - there are no $e, f \in C$ with $e \# f$,
 - if $e \in C$ and $f \preceq e$ then $f \in C$.
- Local configuration $[e] = \{f : f \preceq e\}$. By $\langle e \rangle = [e] - \{e\}$ we denote the set of causal predecessors of e .
- $C \oplus F$ means $C \cup F$ is a configuration and $C \cap F = \emptyset$.
- $Cut(C) = C^\bullet - \bullet C$ gives $state(\mathcal{M})$ the global state of \mathcal{M} after executing the events in C .

Where to cut branching processes?

Def [Cutting context]: It is a pair $\theta = (\sim, \triangleleft)$ such that:

- \sim is an equivalence relation on $Conf$.
- \triangleleft is a strict well-founded partial order on $Conf$ extending \subsetneq relation.
- The two are preserved by finite extensions: if $C \sim C'$ and $C \oplus F$ is defined then there is F' such that
 - + $C \oplus F \sim C \oplus F'$, and
 - + if $C' \triangleleft C$ then $C' \oplus F' \triangleleft C \oplus F$.

Example: $C \sim C'$ if both determine the same marking.

Def: A prefix of $\llbracket \mathcal{M} \rrbracket$ defined by a set of events E_{cut} is **complete** with respect to θ if for every $C \in Conf$ there is $C' \in Conf$ such that $C' \cap E_{cut} = \emptyset$ and $C \sim C'$.

● Assume we have a cutting context $\theta = (\sim, \triangleleft)$. As \triangleleft is a well order we can use it to define *fsble* and *cut* events in $\llbracket \mathcal{M} \rrbracket$.

Def:

$e \in fsble$ if $\langle e \rangle \cap cut = \emptyset$.

$e \in cut$ if $e \in fsble$ and \exists event f with $[f] \triangleleft [e]$, $[f] \subseteq fsble - cut$ and $[f] \sim [e]$.

Prop: For every event e of \mathcal{M} :

$e \in fsble$ iff $\langle e \rangle \in fsble - cut$.

if $e \in cut$ then $e \in fsble$.

Def: Canonical prefix $Pref(\mathcal{M})$ is defined by keeping events up to *cut*.

Thm: Canonical prefix is complete.

- For a configuration C let $Tr_i(C)$ be the sequence of global transitions involving process i .
- Let $V(C) = (Tr_1(C), \dots, Tr_n(C))$.
- We can compare sequences using:
 - size quasi order,
 - lexicographic order,
 - silex order (first size then lexicographic).
- Two adequate orderings:
 - ⋈₁ Lexicographic extension of silex ordering.
 - ⋈₂ First compare by lexicographic extension of size order, if the same then by lexicographic order.
- We need to show: if $C \sim C'$ and $C \triangleleft C'$ then for every F there is F' with $C \oplus F \sim C' \oplus F'$ and $C \oplus F \triangleleft C' \oplus F'$.

Lemma [König lemma for branching systems]: A branching system is finite iff it does not contain an infinite \preceq chain of events.

Cor: The canonical prefix is finite iff \sim is of finite index.

Rem: Canonical prefix can be also finite for nets with infinitely many reachable markings. The problem may lay in finding an adequate ordering.

Prop: If \triangleleft is total then there are at most $|Conf/\sim|$ events in the canonical prefix.

Rem: Actually there are no more than $|Conf_{loc}/\sim|$ configurations.

How to do model checking?

- Take a formula α over subset of the set of processes.
- Translate $\neg\alpha$ into an automaton \mathcal{A} .
- Make a product of \mathcal{M} and \mathcal{A} .
- This construction kills big part of advantages of unfolding.
- There may be still some concurrency left as the automaton sequentializes only the processes it reads.

- Of course it would be better to use Trace LTL.
- How to compile Trace LTL to Petri Nets in an efficient way?
- How to write a PN gadget checking $X_a < X_b$?

- Canonical Prefixes of Petri Net Unfoldings, Khomenko, V., Koutny M. and Vogler W. Acta Informatica, Vol. 40, pp 95-118, 2003
- J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. Formal Methods in System Design, 20:285-310, 2002.
- J. Esparza and S. Römer. An unfolding algorithm for synchronous products of transition systems. In Proc. of CONCUR'99, number 1664 in LNCS, pages 2- 20. Springer-Verlag, 1999
- J. Esparza and K. Heljanko. A new unfolding approach to LTL model checking. ICALP 2000, LNCS 1853, pages 475-486.
- J.-M. Couvreur and S. Grivet and D. Poitrenaud, Unfolding of Products of Symmetrical Petri Nets., ICATPN 2001, LNCS 2075, 121–143
- V. Diekert, P. Gastin, Pure future local temporal logics are expressively complete for Mazurkiewicz traces, LATIN'04, LNCS, to appear 2004