Some applications of Petri Nets to the Analysis of Parameterised Systems

Javier Esparza

Institute for Formal Methods in Computer Science

University of Stuttgart

(with thanks to Jean-Francois Raskin)

Automatic verification

Initiated in the midlle 80s

Very successful in hardware

Application to software systems is probably today's main research challenge

Explicit construction of the state space $\longrightarrow$ finiteness constraint

Sources of infinity in infinite-state systems

Data manipulation: unbounded counters, integer variables, lists . . .

Control structures: procedures , process creation . . .

Asynchronous communication: unbounded FIFO queues

Parameters: number of processes, of input gates, of principals, of sessions, of nonces . . .

Real-time: discrete or dense domains

Parameterised protocols

Defined for n processes.

Correctness: the desired properties hold for every n

Processes modelled as communicating finite automata

Turing powerful, and so further restrictions sensible/necessary

Protocols with anonymous agents

Finite number of process types

- senders and receivers
- readers and writers
- honest principals, intruders, trusted parties

All processes of the same type execute the same algorithm,
i.e., all finite automata of this type are identical

Processes are anonymous (no IDs)

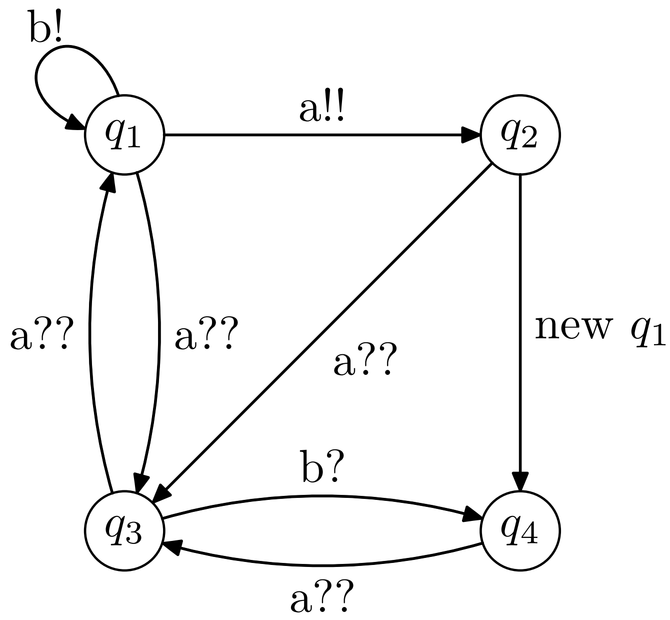
Finite number of messages

Process creation allowed

Communication mechanisms:

- Rendezvous: two processes exchange a message and move to new states
- Bounded fifo channels
- Unbounded channels if overtaking (or loss) possible
- Broadcasts: a process sends a message to all others
  all processes move to new states

Syntax



a!!	:	broadcast a message along (channel) *a*
a??	:	receive a broadcasted message along *a*
b!	:	send a message to one process along *b*
b?	:	receive a message from one process along *b*
new *q*	:	create a new process with initial state *q*

Remark: finite datatypes can be simulated

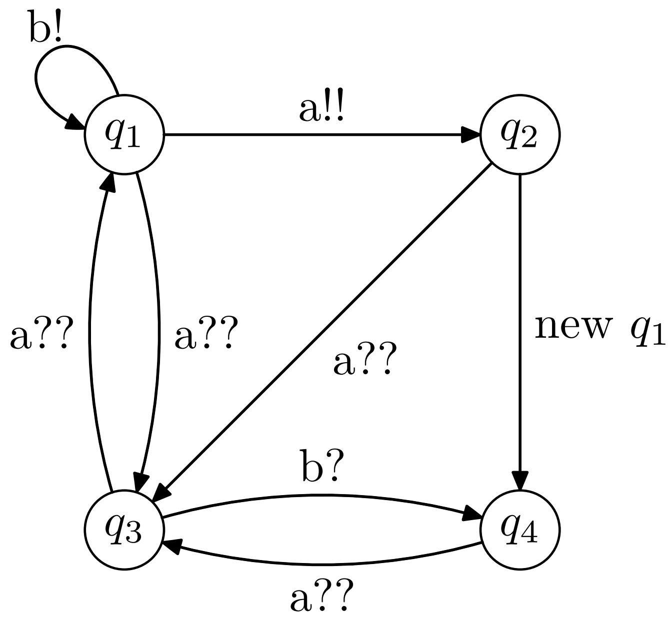
Semantics

The global state of a broadcast protocol is completely determined by the number of processes in each state

Configuration: mapping $c : Q \longrightarrow \mathbb{N}$
represented by the vector $(c(q_1), \ldots, c(q_n))$

Language $L(i)$ for an initial configuration i:
Set of sequences σ such that $i \xrightarrow{\sigma} c$ for some configuration c

ω-language $L_\omega(i)$ for an initial configuration i:
Set of infinite sequences σ such that $i \xrightarrow{\sigma}$



$$(3,1,2,4) \xrightarrow{\quad b \quad} (3,1,1,5) \quad \text{(rendezvous)}$$

$$(3,1,2,4) \xrightarrow{\quad a \quad} (2,1,7,0) \quad \text{(broadcast)}$$

$$(3,1,2,4) \xrightarrow{\;new\ q_1\;} (4,0,2,5) \quad \text{(process creation)}$$

ω-semantics

ω-configuration: mapping $C : Q \to \mathbb{N} \cup \{\omega\}$

Intuition for $C(q) = \omega$: arbitrarily many processes on q

Intuition for C: set of configurations obtained replacing ωs by arbitrary numbers

Formalization using abstract interpretation

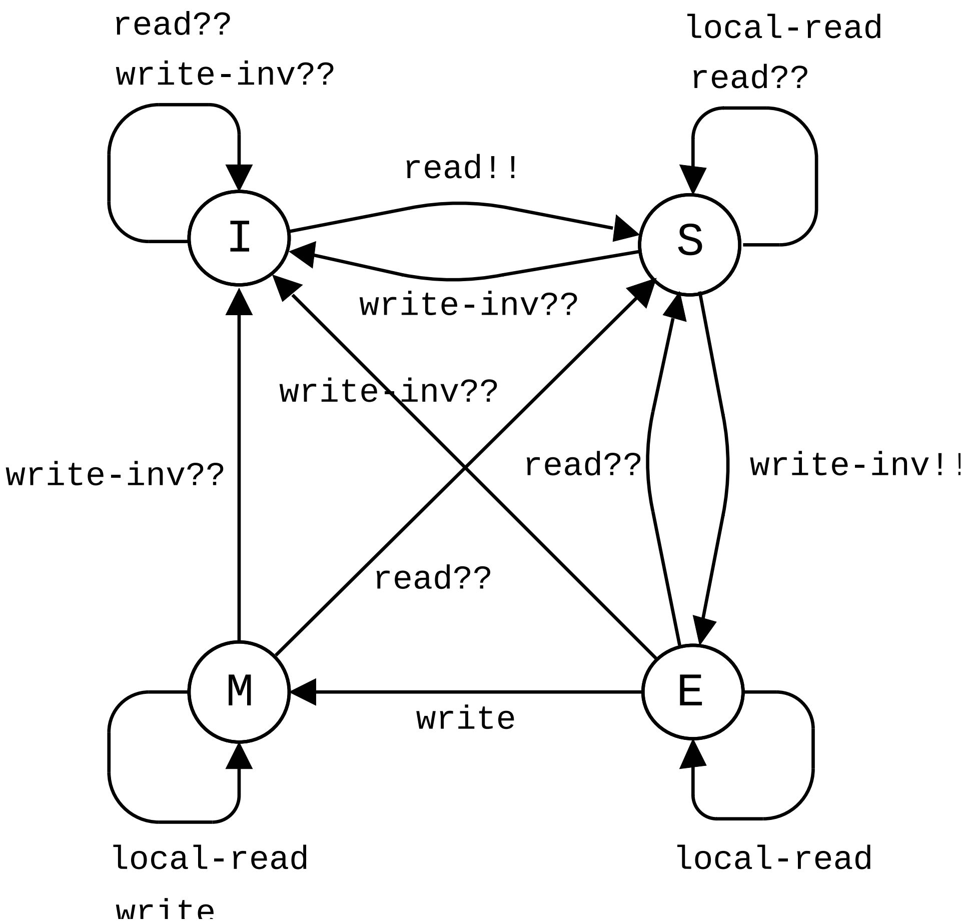
Language $L(I)$ for an initial ω-configuration I:

$$L(I) = \bigcup_{c \in I} L(c)$$

ω-language $L_\omega(I)$ for an initial ω-configuration I:

$$L_\omega(I) = \bigcup_{c \in I} L_\omega(c)$$

A MESI-protocol



Connection to Multi-Transfer-Petri nets

States $\longrightarrow$ places

ω-configurations C $\longrightarrow$ ω-markings M (set of ordinary markings)

Rendezvous, process creation $\longrightarrow$ ordinary transitions

Broadcast $\longrightarrow$ multi-transfer transitions

- pairs of input and output transfer arcs
  (several pairs may share the output arc)
- input arc removes all tokens from input place (possibly 0!!)
- output arc adds same number of tokens to output place

Each transition t has attached a linear transformation T_t

$$T_t(X) = A_t \cdot X + b_t$$

where A nonnegative, such that if $M \xrightarrow{t} M'$ then $M' = T_t(M)$

– $\omega + n = \omega \cdot n = \omega, \quad \omega + \omega = \omega$

– If t models rendezvous or *new q* , then A_t is the identity

– If t models broadcast, then A_t is a 0-1 matrix with unit vectors as columns

$$\begin{pmatrix} m'_1 \\ m'_2 \\ m'_3 \\ m'_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Consequence: for every sequence σ of transitions, there is also a linear transformation T_σ that computes the effect of the sequence

Verification problems

Safety

Given: a multi-transfer net (N, I), a regular language D of dangerous finite transition sequences.
To decide: if $L(I) \cap D = \emptyset$.

Liveness

Given: a multi-transfer net (N, I), a regular language D, an ω-regular language D of dangerous infinite behaviours.
To decide: if $L_\omega(I) \cap D = \emptyset$.

Reduction

Using the automata-theoretic approach to model-checking the safety and liveness problems are reduced to

– Coverability

  Given: a multi-trasfer net (N, I), a marking f (no ωs)
  To decide: if f can be covered from I, i.e. if there exists σ such that $I \xrightarrow{\sigma} M \geq f$, where $\geq$ is the pointwise order with $\omega > n$ for all numbers n

– Repeated coverability

  Given: a multi-trasfer net (N, I), a marking f (no ωs)
  To decide: if f can be repeatedly covered from I, i.e. if there exists an infinite run from I which covers f infinitely often

Forward search

Construct the reachability graph according to the ω-semantics starting from I

If some node of the graph covers f, answer 'coverable'

Problem: non-termination
(even for the normal semantics)

Karp-Miller's acceleration

Karp and Miller, 69, German and Sistla, JACM 39(3), 92

if $M_1 \xrightarrow{\sigma} M_2$ and $M_1 \leq M_2$ then replace M_2 by the *lub* w.r.t. $\leq$ of the chain

$$M_1 \xrightarrow{\sigma} M_2 \xrightarrow{\sigma} M_3 \xrightarrow{\sigma} \ldots .$$

Since

$$M_1 \xrightarrow{\sigma} M_2 \xrightarrow{\sigma} M_3 \xrightarrow{\sigma} \ldots$$

is equal to

$$M_1 \xrightarrow{\sigma} T_\sigma(M_1) \xrightarrow{\sigma} T_\sigma^2(M_1) \xrightarrow{\sigma} \ldots$$

M_2 is replaced by

$$lub\{T_\sigma^n(M_1) \mid n \geq 0\}$$

Basic property: if $M_1 \xrightarrow{\sigma} M_2$ then $lub\{T_\sigma^n(M_1) \mid n \geq 0\}$ is coverable from M_1

Questions:

How can $lub\{T_\sigma^n(M_1) \mid n \geq 0\}$ be computed ?

Does the acceleration guarantee termination ?

Computing *lubs*

Place/Transition nets:

$$T_\sigma(M_1) = M_1 + b_\sigma$$

$$lub\{T_\sigma^n(M_1)\} = M_1 + \omega \cdot b_\sigma$$

Multi-transfer nets (Emerson, Namjoshi LICS 98):

$T_\sigma(M_1) = A_\sigma \cdot M_1 + b_\sigma$, where A_σ 0-1 matrix with unit vectors as columns

There is $i < j$ such that $A_\sigma^i = A_\sigma^j$

$$lub\{T_\sigma^n(M_1)\} = A_\sigma^i(M_1) + \sum_{k\in[0,i)} A_\sigma^k(b_\sigma) + \omega \cdot \sum_{k\in[i,j)} A_\sigma^k(b_\sigma)$$

This may take exponential time in the number of states

Termination

Place/Transition nets: guaranteed (Karp, Miller, 69)

Assume $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} M_3 \xrightarrow{t_3} \ldots$ in coverability graph

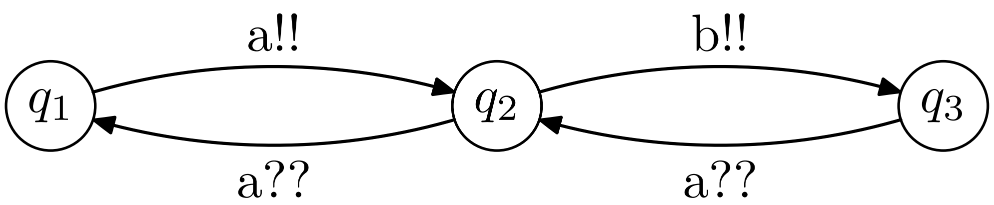
By Dickson's lemma we find i, j with $M_i \xrightarrow{\sigma} M_j$ and $M_i \leq M_j$, $M_i \neq M_j$

Replacing M_j by $M_i + \omega \cdot b_\sigma$ adds at least one ω to M_i

ω never goes away

Contradiction!

Multi-transfer nets: not guaranteed (E., Finkel, Mayr LICS 99, Finkel, Leroux 00)



The sequence $abab^2ab^3ab^4 \ldots$ 'survives' the acceleration

Conclusions

Karp-Miller acceleration adequate for place/transition nets

Non-primitive recursive size in the worst case!

However, asymptotically optimal EXPSPACE algorithm much worse in practice

Serious problems for multi-transfer nets

Termination fails in very simple cases

Computation of *lubs* complicated

Searching backwards

Let F be the set of markings that cover f

f is coverable from I iff F is reachable from I

Backward search

$pre(M)$ = immediate predecessors of M

Initialize $M := F$

Iterate $M := M \cup pre(M)$ until

$M \cap I \neq \emptyset$; return "coverable", or

a fixpoint is reached; return "non-coverable"

Question: When is the procedure effective?

Backward search effective if . . .

Backward search is effective if there is a class $\mathcal{C}$ of sets of markings satisfying Conditions (1) - (6) below

1. each $M \in \mathcal{C}$ has a symbolic finite representation
2. $F \in \mathcal{C}$
3. if $M \in \mathcal{C}$, then $M \cup pre(M) \in \mathcal{C}$ (and effectively computable)
4. emptyness of $M \cap I$ is decidable
5. $M_1 = M_2$ is decidable (to check if fixpoint has been reached)
6. any chain $M_1 \subseteq M_2 \subseteq M_3 \ldots$ reaches a fixpoint after finitely many steps

(1) - (5) guarantee partial correctness, (6) guarantees termination

Upward-closed sets

A set M of markings is upward-closed if

$$m \in M \text{ and } m' \geq m \text{ implies } m' \in M$$

Conditions (1)-(6) hold for the class of upward-closed sets

General principle: Abdulla, Cerans, Jonsson, and Tsay, I&C 160, 2000

Application to broadcasts (transfer nets): E., Finkel, Mayr, LICS'99

1. An upward-closed set can be finitely represented by its set of minimal elements w.r.t. the pointwise order $\leq$

3. If M is upward-closed then so is $M \cup pre(M)$

   Since union of upward-closed sets is upward-closed, it suffices to prove that $pre(M)$ is upward-closed

   Take $m \in pre(M)$ and $m' \geq m$. We show $m' \in pre(M)$

$$
\begin{array}{ccc}
m & \xrightarrow{t} & l \in M \\
\leq & & \leq \\
m' & \xrightarrow{t} & l' \in M
\end{array}
$$

6. Any chain $M_1 \subseteq M_2 \subseteq M_3 \ldots$ of upward-closed sets reaches a fixpoint after finitely many steps.

Conclusions

Backwards search on upward-closed sets is guaranteed to terminate for multi-transfer nets, even for nets with arbitrary non-negative linear transformations

Implementation very similar for place/transition and transfer nets

Repeated coverability

Place/transition nets: decidable

Construct the coverability graph

Find ‘pumpable sequence’ containing a node that covers f

Multi-transfer nets: undecidable

Proof: By reduction from the halting problem for counter machines

The ‘pumpable sequence’ above still works, but coverability graph may now be infinite

Weak simulation of counter machines by transfer nets

Counter machine	Transfer net
$q \xrightarrow{c := c+1} q'$	$(q, \mathit{Store}) \xrightarrow{inc_c} (q', c)$
$q \xrightarrow{c := c-1} q'$	$(q, c) \xrightarrow{dec_c} (q', \mathit{Store})$
$q \xrightarrow{c = 0} q'$	$(q, c) \xrightarrow{reset_c} (q', Sink)$

Cheat: $reset_c$ transition executed with tokens on c

Let N be the total number of tokens in the counters and the *Store*

The argument

N does not increase along an execution of the net,

and it decreases if and when the protocol cheats.

$\Longrightarrow$ No infinite execution cheats infinitely often.

$\Longrightarrow$ Infinite executions are ultimately honest.

$\Longrightarrow$ The net has an infinite execution iff

the counter machine has an infinite run.

Backward search in practice

Backwards search computes non-reachable states $\longrightarrow$ 'Set explosion' problem

Solutions by Bozzano, Delzanno, Raskin, et al. (several papers)

Use dedicated data structures to compactly represent (upward-closed) sets of markings

Use place invariants to prune unreachable markings

Data structures

Dags with integers as nodes

- Each path through the dag represents one marking
- Equality test is coNP-complete

Linear constraints (Delzanno, E., Podelski, CSL99)

- Constraints of the form $x_1 + \ldots + x_n \geq c$
- *pre* amounts to computing a linear transformation
- Equality check is coNp-complete
- Introduce weaker equality check: larger number of iterations, but each check can be performed in polynomial time

Place invariants

Markings violating the invariant are unreachable

Basic idea: intersect the current upward-closed set with the invariant

Problem: upward-closure gets lost

Solution: remove only minimal elements whose upward-closure does not intersect the invariant

Some experiments by Delzanno, Raskin, et al.

Cache coherence protocols, communication protocols (around 10-20 examples)

Model sometimes needs to be extended

Some dozens of places and transitions

Verification of simple safety properties (mutual exclusion, reachability)

Success in most cases

Verification time: mostly seconds, sometimes minutes

Memory consumption: from a few to some dozens of MB

Conclusions

Verification for infinite families of systems is possible

Naïve basic algorithms are easy to implement

However, good data structures and heuristics are essential

Difficult trade-off: expressivity vs. efficiency