Describing Easy Symmetries of Petri nets Compositionally

Bernard Berthomieu[1,2] ...

[1] CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France
[2] Univ de Toulouse, LAAS, F-31400 Toulouse, France

1 Introduction

symmetries reduction a well know technique for fighting comb explosion in model-checking

used in different forms in various contexts ...
for time petri nets, originating from Starke ...
two phases: identifying symmetries (1) and checking equivalence (2) ...
schmidt/lola approach: (1) automatic, (2) by ...

(1) reasonnably fast in e.g. lola, (2) may be slow ...
pro: (1) finds all symmetries, so great potential reduction
con: but (2) rely on approximate "minimization", sp some of the benefits of (1) lost

alternative for (1): symmetry annotations make explicit (1)'
alternative for (2): computation of representants (2)'
we present an approach combining (1)' and (2)':

(1)' by structural, compositional, description of symmetries, with local restrictions ensuring polynimial COP
(2)' relies on sorting methods.
pro: really fast, (2)' way faster than
con: not all symmetries can be expressed or handled, and they must be made explicit

justif: large PN have some structure, known by the designer, and symmetries known too.

2 describing Petri nets (structurally ?)

a basic net is a place with its connected edges and transitions, or a single transition

compound nets are built by composition by |

Define |

expressiveness: prove that any net can be built that way

enriching the language (for convenience, same expressiveness): labels, renaming, free product ?, ...
examples (structure explicit ...)

3 compositional description of symmetries

3.1 idea

assuming subnets, each with its symmetries (computed or declared), infer symmetries on result of composition.
method, how to compute compositons of symmetries ?

automorphismes et groupes rsultant des compositions ? rapport avec calcul de schmidt
benefits, does it help computing generators ?

expressiveness

not all automorphisms can be computed this way $(A(g_1)|A(g_2)) \subseteq A(g_1|g_2)$
products have more symmetries than made explicit (e.g. identical components, dihedral, or pool of pools ...)

3.2 Products

Given $N_1 :: G_1$ and $N_2 :: G_2$, what are $N_1 \mid N_2$ and $G_1 \mid G_2$?
Define $N_1 \mid N_2$; semantics in terms of lts
Define $G_1 \mid G_2$
semantics: $G_1 \mid G_2$ should capture all symetries of G_1 and G_2, but no more.
computation of generators
soundness
"completness" (in above sense).
expressiveness: clearly, there can be more symmetries in $N_1 \mid N_2$ that expressed in $G_1 \mid G_2$, even if G_i captures all symmetries of N_i

condition of starke on transitions
note: compositions preserve isomorphism of components
algorithmics: not polynomial in general, next section concentrate on restrictions ensuring polynomial COP

4 easy symmetries, compositional description of

4.1 Symmetries having polynomial COP (cf. Donaldson)

Pool, Ring, disjoint product, wreath product, ++

4.2 Specifying symmetries – symmetry annotations

Pool(n,N::G) and Ring (n,N::G) and compositions of

semantics: what is it meant by Pool or Ring ?
Pool (resp. Ring): symmetries are those resulting by process swaps (resp. circular process permutations)

Computation of generators:

Pool/Ring (n,N::G):
if G = Ident:
    generators from isomorphisms ?
    checking that they indeed define a symmetric (resp. cyclic) group
    conditions for "easyness"
if G $\neq$ Ident :
    genarators ?
    semantics ?

4.3 Easyness conditions

"easy" $\Leftrightarrow$ COP is polynomial
1. When does a Pool or Ring describe an "easy" symmetry ?
condition of "moves" of components of a pool or ring. If satisfied, then lexmin in orbit can be computed by sorting component states
condition in untimed and timed cases
basic components partition places $\Rightarrow$ trivially easy in untimed
if timed:
2. products: when is result a disjoint product ?
condition in untimed and timed cases (based on moves and computation of generators of the product in prev. section)
3. embedded symmetries:
Pool/ring of Pool/Ring: when is result a wreath product ?
condition in untimed and timed cases

4.4 Extensions

Generalizations: Cube ? Dihedral ? Open groups with adhoc COP ?

Q: reconstructing “easy” group structure from computed symmetries “ la lola” ?

5 computing canonical state representants

adapt standard result to petri nets

what if net timed (in discrete time)

if untimed: “moved” condition applies to places only
if times then “moved” condition applied to transitions as well sine the tme information is captured by transitions

computing experiments

examples

comparison with lola and ina (and Junttila ?)

what if net timed (in discrete time)

what if disjointness conditions between pool members not satisfied ? We don’t obtain canonical forms, but do we obtain minimal forms anyway ? (sound ? termination ?)

6 conclusion

orthogonal arguments, pro/con of each

merging both approaches

identifying easy symmetries in lola style treatment whenever possible
???

A States

A.1 Of Petri nets

$P = \{p_1, \ldots, p_n\}$ is the set of places, assumed ordered.
$T = \{t_1, \ldots, t_m\}$ is the set of transitions, assumed ordered.
$s = m_1, \ldots, m_n$
where m_i is the marking of place p_i

states can be ordered by $\leq$ = lex ordering on $\mathbb{N}^n$

A.2 Of Time Petri nets

In discrete time – clocks

$s = m_1, \ldots, m_n, k_1, \ldots, k_m$
where m_i is the marking of place p_i
and k_j is the clock value of transition t_j ($\perp$ if not enabled, conventionally)

let $\mathcal{N} = \mathbb{N} \cup \{\perp\}$ and assume $(\forall x \in \mathbb{N})(\perp \leq x)$
states can be ordered by $\leq$ = lex ordering on $\mathcal{N}^{n+m}$
or by colex ordering on $\mathcal{N}^n \times \mathcal{N}^m$, seeing states as pairs $(m_1, \ldots, m_n), (k_1, \ldots, k_m)$

In dense time – DBM

DBM encodes system:

$\alpha_i \leq t_i \leq \beta_i$
$t_i - t_j \leq \gamma_{i,j}$

by:

$d_{0,0} = 0$ (conventionally)
$d_{0,i} = -\alpha_i$ $(i \neq 0)$
$d_{i,0} = \beta_i$ $(i \neq 0)$
$d_{i,j} = \gamma_{i,j}$ $(i, j \neq 0, i \neq j, \infty$ added ....)

Then:
$s = m_1, \ldots, m_n, d_{0,0}, \ldots, d_{0,m}, \ldots, d_{m,0}, \ldots, d_{m,m}$
== linearized DBM. Note: more state entries than transitions

Explain effects of a permutation on the state.
e.g. swapping transitions i and j in the dbm $(i, j \neq 0)$:

$d_{i,j} \leftrightarrow d_{j,i}$
$(\forall k)(d_{i,k} \leftrightarrow d_{j,k})$
$(\forall k)(d_{k,i} \leftrightarrow d_{k,j})$

ordering states: colex on $\mathcal{N}^n \times \mathcal{N}^{(m+1)^2}$

In dense time – ranks

$s = m_1, \ldots, m_n, r_1 \ldots, r_m$
where m_i is the marking of place p_i
and r_j is the rank of variable (transition) t_j in the DBM ($\perp$ if not enabled)

ordering states: as for clocks

B Register

B.1 Model

We wish to model a register with values in range $\{1, \ldots, n\}$

One place per value: 1, ..., n

transition i_j changes value of register from i to j

Initial state is established non-deterministically using a place 0 and transitions 0_i from place 0 to place i

B.2 symmetries

process swaps

symmetries: prove that isomosphic to S_n

observe that, if seen as a TPN, swapping processes i and j modifies the states of other processes as well ...

In terms of cycles, this can be formulated as ...

Show that canonization by sorting, using clocks or ranks, would not necessarily lead to the canonical state (there are local minima)

How to handle such symmetries ? restrictions to ensure that process swaps are " legal"

there are several isomorphisms between components i and j, actually ...

B.3 in tpn+

Q: how to express these symmetries in tpn+sym (using renaming ?)

Pool(n,R,N) ?

condition on R ensuring symmetry is a Sym ?

renaming R % scalarsets