

LTL model checking and some optimizations

Jean-Michel Ilié
Denis Poitrenaud
MoVe – LIP6

Verification

*High-Level
language
with an
operational
semantics*

Formal
specification of
a finite system

Formal
specification of
a property

*Temporal
logic*

Verification
algorithm

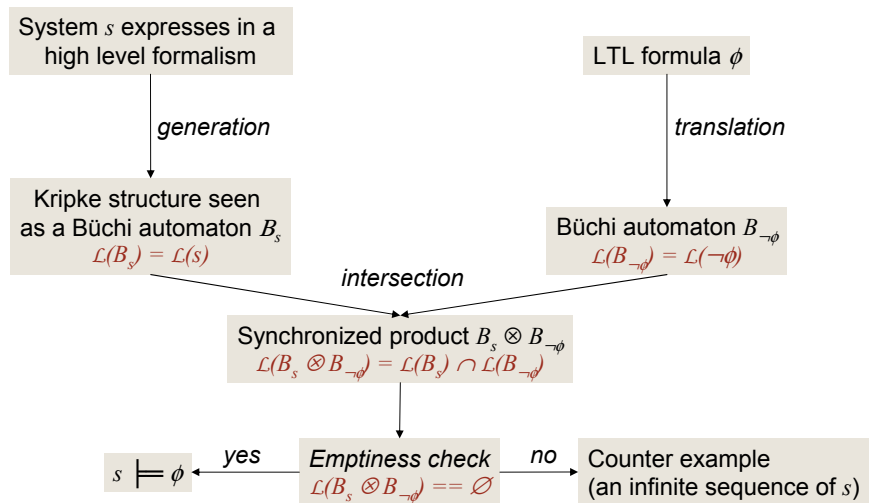
OK

Counter
example

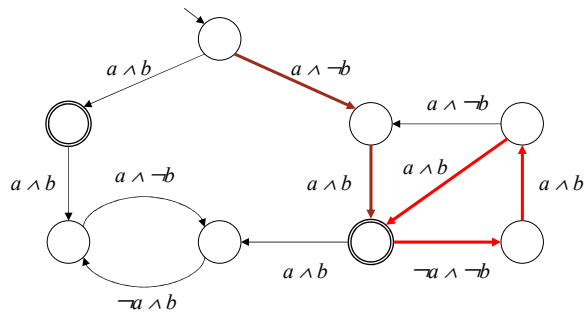
LTL Verification

- LTL addresses only *path property*
each maximal sequence of the system must satisfy the formula
- Expressivity “more” natural than CTL
reachability, safety, liveness and fairness
- Verification algorithms “less” efficient than CTL ones
- Two classical approaches :
 - By automata (also called *explicit approach*) – *Spin, Spot, etc*
 - By set (also called *symbolic approach*) – *NuSMV, Cadence SMV, etc*

Explicit approach

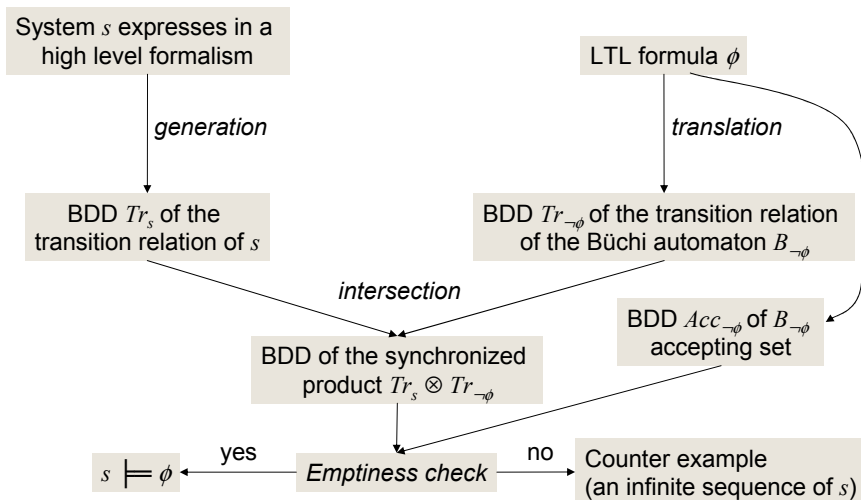


Emptiness check of an explicit Büchi automaton

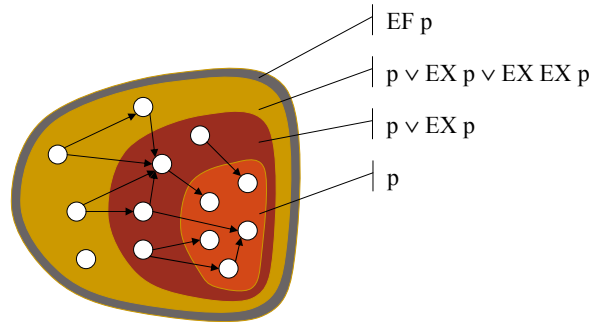


- Look for an infinite path visiting infinitely often an accepting state
- The search is performed on-the-fly (the synchronized product is constructed on demand and the search is stopped as soon as a counter example is found)

Symbolic approach



Emptiness check of a symbolic Büchi automaton



- The check is reduced to the verification of a CTL formula *with fairness constraint*
- The search is performed by successive fixed-point computations (back-tracking the transition relation)
- The search cannot be performed on-the-fly

Explicit versus Symbolic

- Both approaches are used in the industry (Spin and NuSMV)
- Each of them can deal with consequent subpart of complex systems
- A recent study (Rozier and Vardi, SPIN'07) shows that symbolic model checkers overcome explicit ones for the problem of satisfiability of an LTL formula
- No serious experimentation has been realized for the model-checking problem

Existing optimizations

Explicit

- Partial order
 - stubborn set
 - sleep set
 - covering step
 - Unfolding
 - ...
- Semi-decision procedures
 - bit state hashing
 - ...
- Symmetries
- ...

Symbolic

- Bounded model-checking (SAT or BDD)
- All optimizations related to decision diagrams
 - Clustering of transition relations
 - Local saturations
 - ...
- ...

Explicit and symbolic

- CEGAR loop
- ...

Our talk

Two optimizations

- Symbolic observation graphs
 - Exploit the stuttering invariance of LTL \setminus X formulas
 - Mix symbolic and explicit approaches
- Symbolic reachability graph
 - Exploit the symmetries of the system to fold the state space
 - Large application domain (LTL, CTL, CTL*, performance evaluation,...)

MC-SOG: An LTL Model Checker based on Symbolic Observation Graphs

Kais Klai ⁽¹⁾ and Denis Poitrenaud⁽²⁾

⁽¹⁾ LIPN, Université Paris 13, France

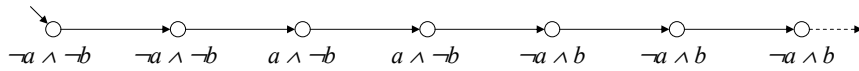
⁽²⁾ LIP6, Université Paris 6, France

Outline

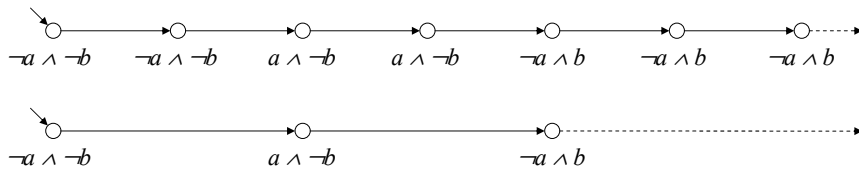
- Introduction
- SOG definition
- SOG implementation – application to Petri nets
- Evaluation
- Conclusion and future work

Stuttering equivalence

- LTL is evaluated on infinite paths of a Kripke structure

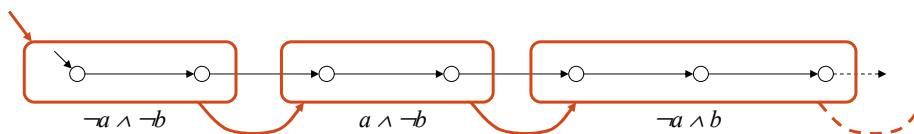


- The evaluation of an LTL\X formula is invariant face to stuttering



Intuition

- Agglomerate adjacent states satisfying the same atomic propositions



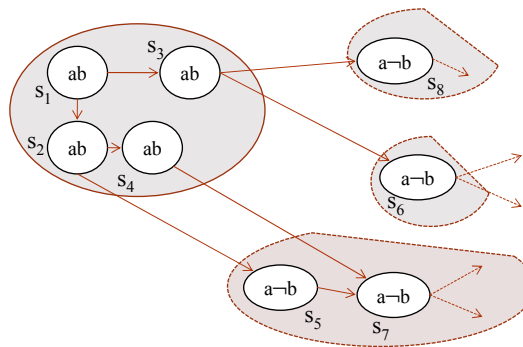
- Problems
 - How to apply this at the graph level ?
 - How to identify has equivalent two such aggregates of states ?
 - How to preserve maximal paths ?
 - Infinite paths
 - Dead paths
 - Divergent paths

Symbolic Observation Graph (SOG)

- An *aggregate* is a set of states satisfying the same atomic propositions
- The preservation of *maximal paths* implies the detection of *deadlock* and *circuits*

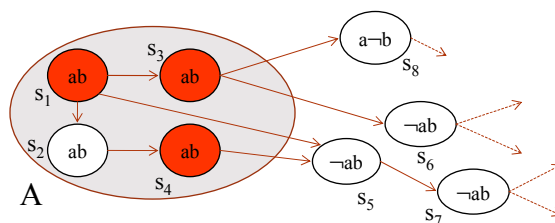


- Symbolic algorithms (based on BDD)
 - To compute and store aggregates
 - To detect deadlock within an aggregate
 - To detect circuit within an aggregate



Notations

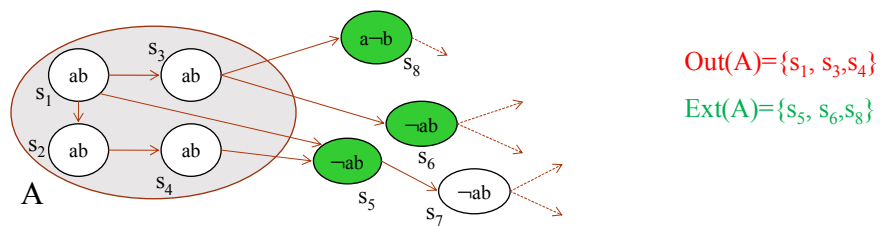
- Let A be an aggregate
 - $Out(A) = \{s \in A \mid \exists s' \notin A \text{ s.t. } s \rightarrow s'\}$



$$Out(A) = \{s_1, s_3, s_4\}$$

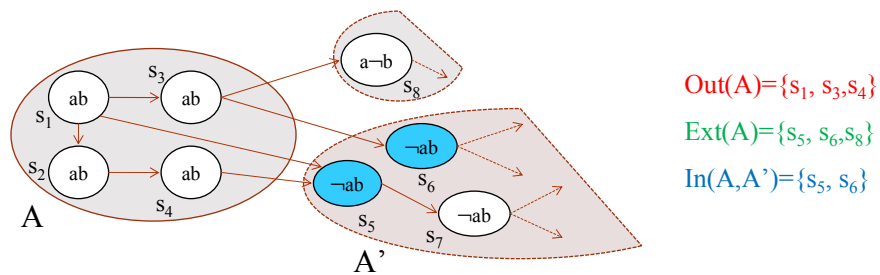
Notations

- Let A be an aggregate
 - $Out(A) = \{s \in A \mid \exists s' \notin A \text{ s.t. } s \rightarrow s'\}$
 - $Ext(A) = \{s' \notin A \mid \exists s \in A \text{ s.t. } s \rightarrow s'\}$



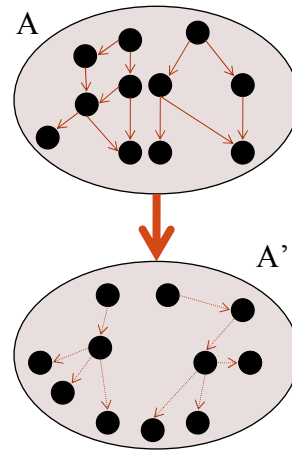
Notations

- Let A be an aggregate
 - $Out(A) = \{s \in A \mid \exists s' \notin A \text{ such that } s \rightarrow s'\}$
 - $Ext(A) = \{s' \notin A \mid \exists s \in A \text{ such that } s \rightarrow s'\}$
- Let A, A' be two aggregates
 - $In(A, A') = Ext(A) \cap A'$



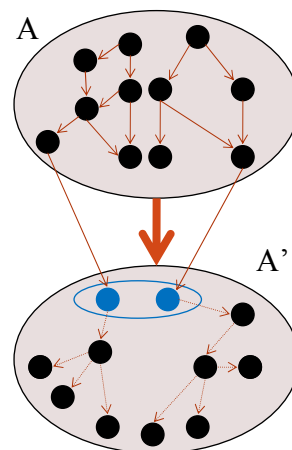
Transition relation of a SOG

- Let A, A' be two aggregates such that $A \neq A'$



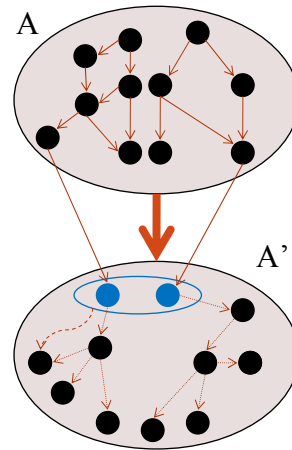
Transition relation of a SOG

- Let A, A' be two aggregates such that $A \neq A'$
- $A \rightarrow A'$ implies
 - $\text{In}(A, A') \neq \emptyset$



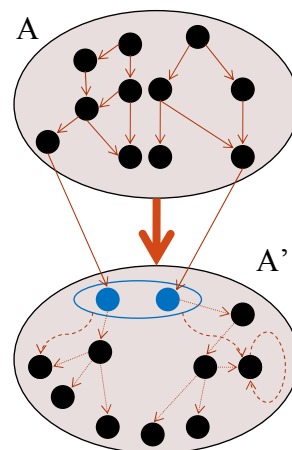
Transition relation of a SOG

- Let A, A' be two aggregates such that $A \neq A'$
- $A \rightarrow A'$ implies
 - $\text{In}(A, A') \neq \emptyset$
 - $\text{Dead}(A') \Rightarrow$ a dead state of A' is reachable from some $s \in \text{In}(A, A')$



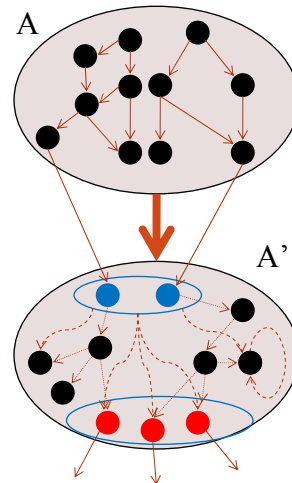
Transition relation of a SOG

- Let A, A' be two aggregates such that $A \neq A'$
- $A \rightarrow A'$ implies
 - $\text{In}(A, A') \neq \emptyset$
 - $\text{Dead}(A') \Rightarrow$ a dead state of A' is reachable from some $s \in \text{In}(A, A')$
 - $\text{Live}(A') \Rightarrow$ a circuit of A' is reachable from some $s \in \text{In}(A, A')$



Transition relation of a SOG

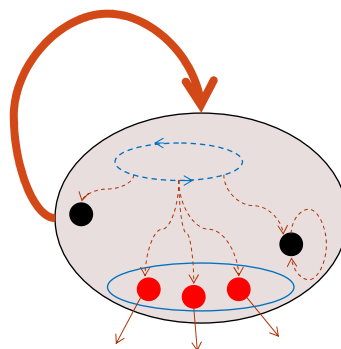
- Let A, A' be two aggregates such that $A \neq A'$
- $A \rightarrow A'$ implies
 - $\text{In}(A, A') \neq \emptyset$
 - $\text{Dead}(A') \Rightarrow$ a dead state of A' is reachable from some $s \in \text{In}(A, A')$
 - $\text{Live}(A') \Rightarrow$ a circuit of A' is reachable from some $s \in \text{In}(A, A')$
 - Each output state of A' is reachable from some $s \in \text{In}(A, A')$



➔ The set $\text{In}(A, A')$ is said to be *compatible* with A'

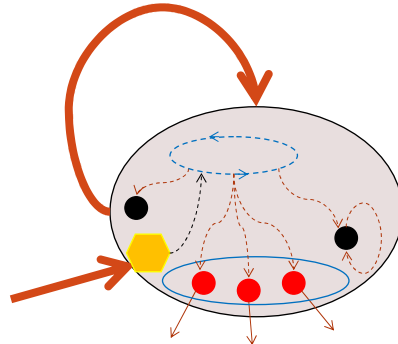
Transition relation of a SOG

- Let A be an aggregate
- $A \rightarrow A$ implies
 - There exists a circuit in A which is compatible with A



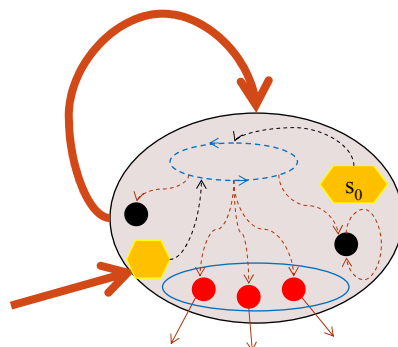
Transition relation of a SOG

- Let A be an aggregate
- $A \rightarrow A$ implies
 - There exists a circuit in A which is compatible with A
 - For each aggregate B , predecessor of A , there exists such a circuit reachable from $\text{In}(B, A)$



Transition relation of a SOG

- Let A be an aggregate
- $A \rightarrow A$ implies
 - There exists a circuit in A which is compatible with A
 - For each aggregate B , predecessor of A , there exists such a circuit reachable from $\text{In}(B, A)$
 - If $s_0 \in A$ then there exists such a circuit reachable from s_0

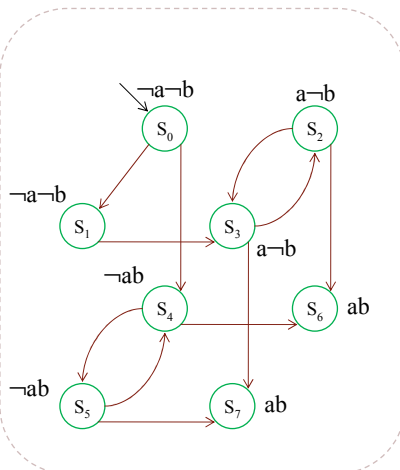


Symbolic Observation Graph (SOG)

Let $T = \langle \Gamma, L, \rightarrow, s_0 \rangle$ be a Kripke structure over a set of atomic propositions AP

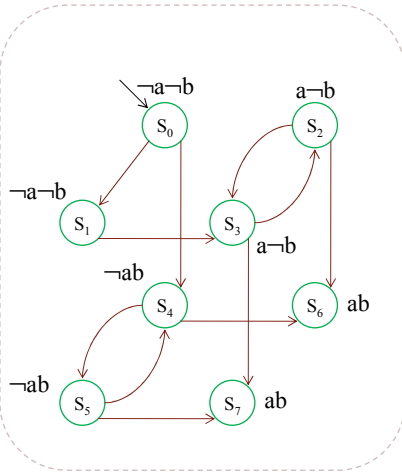
- A SOG of T is a tuple $G = \langle \Gamma', L', \rightarrow', a_0 \rangle$ where:
 - Γ' is a set of *aggregates*
 - $L' : \Gamma' \rightarrow 2^{AP}$ is a labeling function
 - $\rightarrow' \subseteq \Gamma' \times \Gamma'$ is the transition relation
 - a_0 is the initial aggregate compatible with $\{s_0\}$

Example of SOG

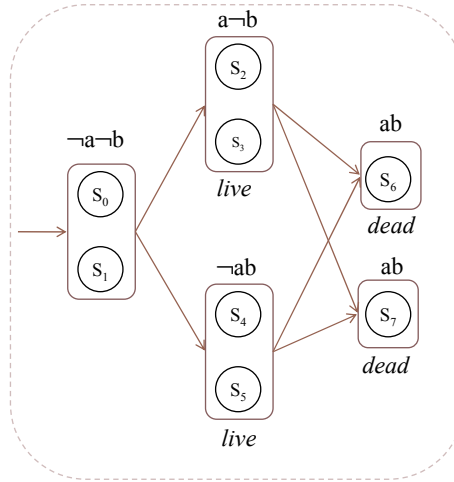


A Kripke structure

Example of SOG

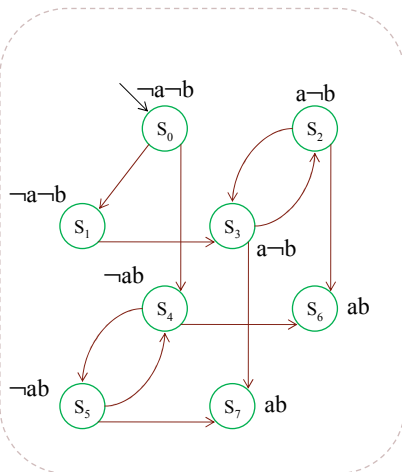


A Kripke structure

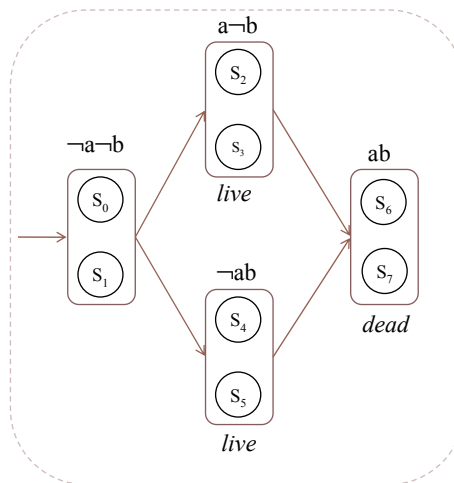


A corresponding SOG

Example of SOG



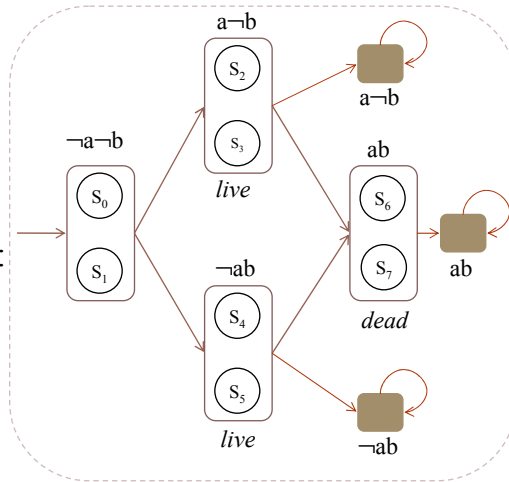
A Kripke structure



Another corresponding SOG

Extended SOG (ESOG)

- Goal: adapt the SOG structure to the automata theory approach of LTL model checking
- For each aggregate A s.t. Dead(A) or Live(A) hold:
 - Add a new state $L'(A)$, labeled with $L'(A)$ (if it does not exist)
 - Add $(A, L'(A))$ to \rightarrow'
 - Add $(L'(A), L'(A))$ to \rightarrow'

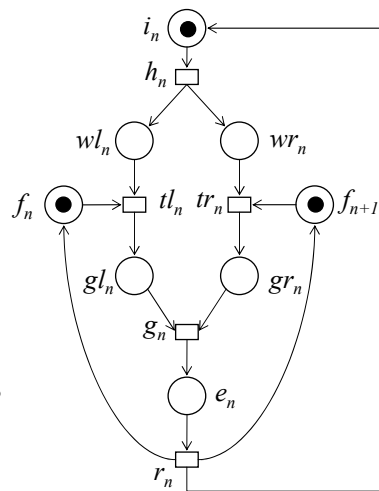


Application to bounded Petri nets

$$\varphi = G((wl_1 \wedge wr_1) \Rightarrow F(e_1))$$

- **Observed transitions:** transitions modifying the marking of place used as propositions
 - $\{h_1, tl_1, tr_1, g_1, e_1\}$
- **Local transitions:** the others
 - $\{h_i, tl_i, tr_i, g_i, e_i\}$ for all $i \neq 1$

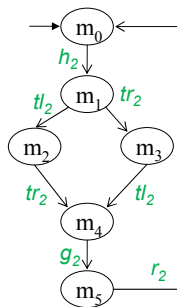
Philosophers model



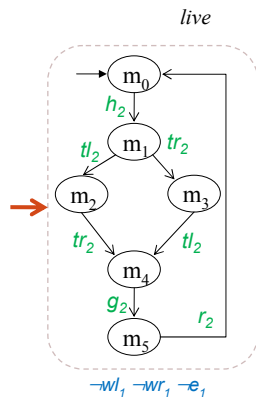
SOG construction for two philosophers



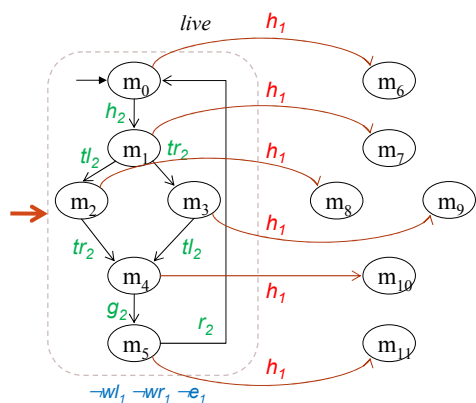
SOG construction for two philosophers



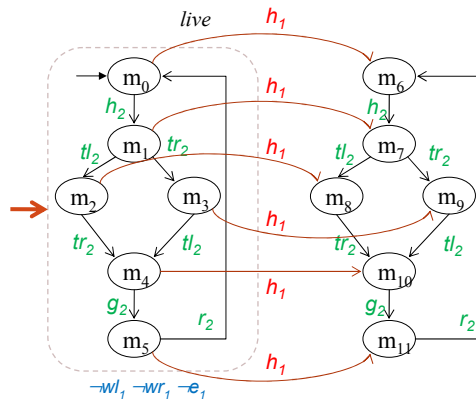
SOG construction for two philosophers



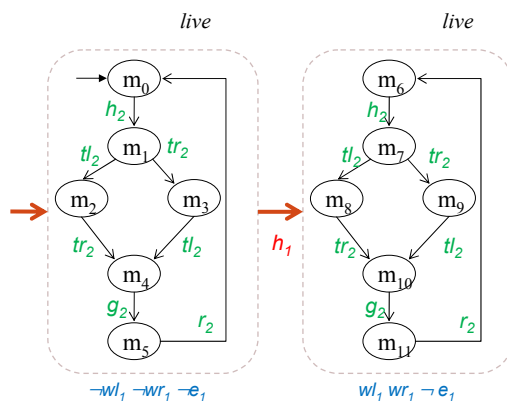
SOG construction for two philosophers



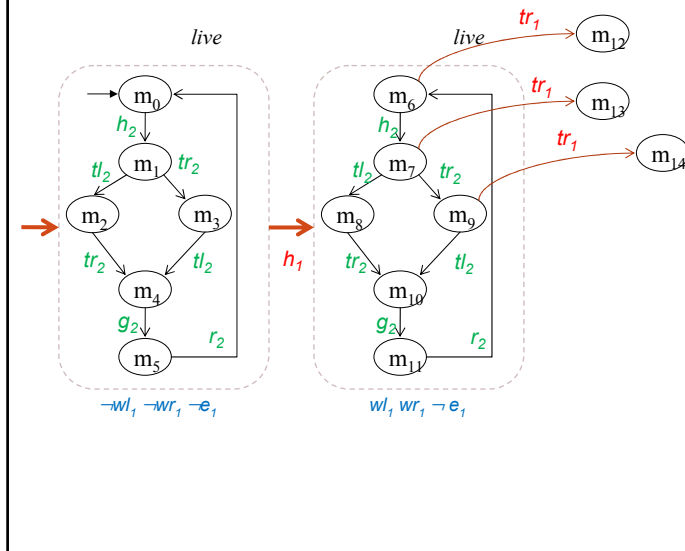
SOG construction for two philosophers



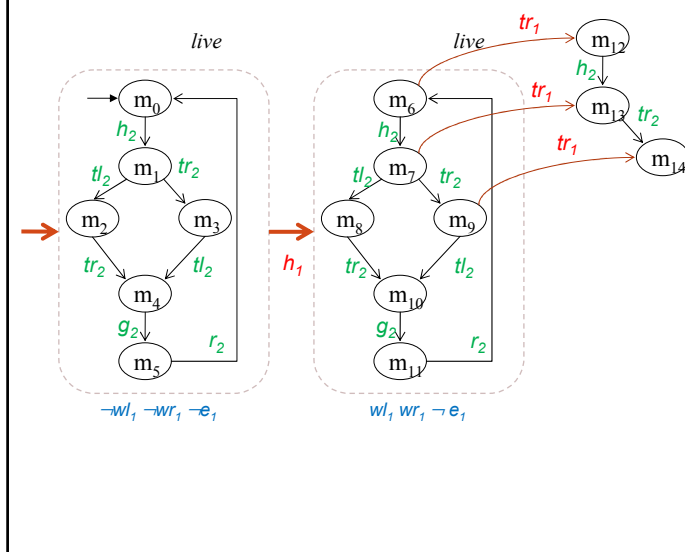
SOG construction for two philosophers



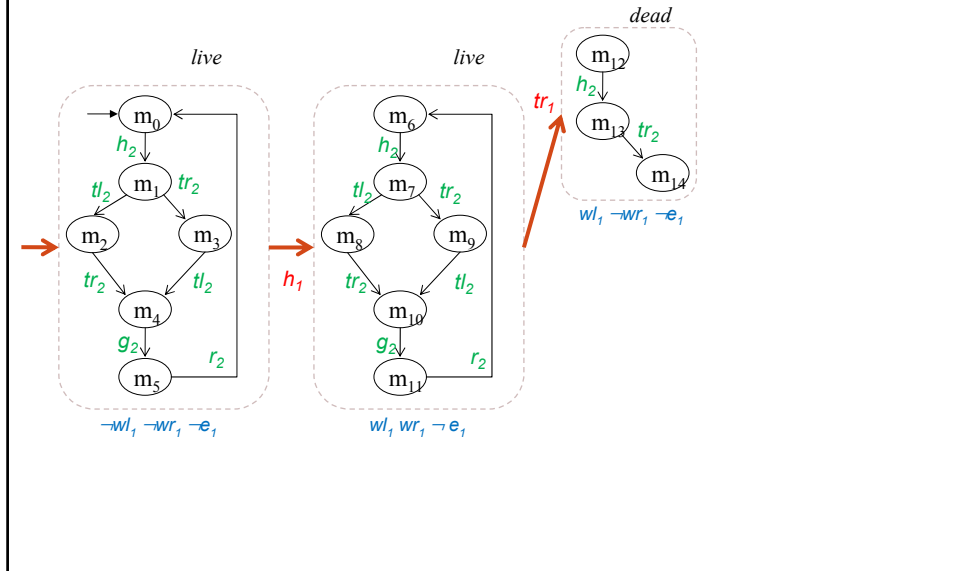
SOG construction for two philosophers



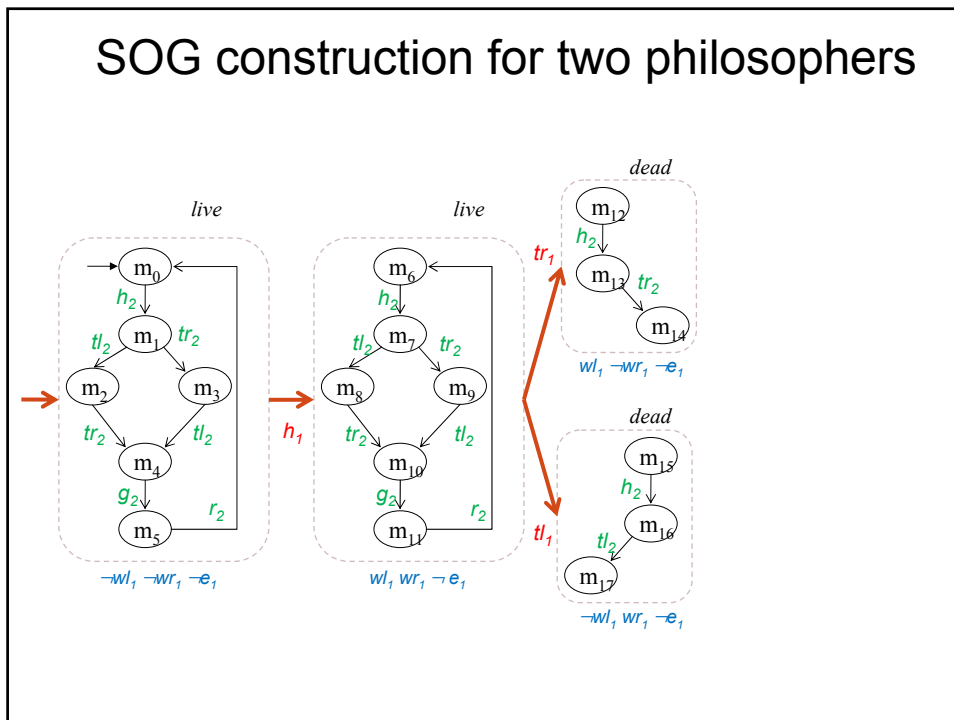
SOG construction for two philosophers



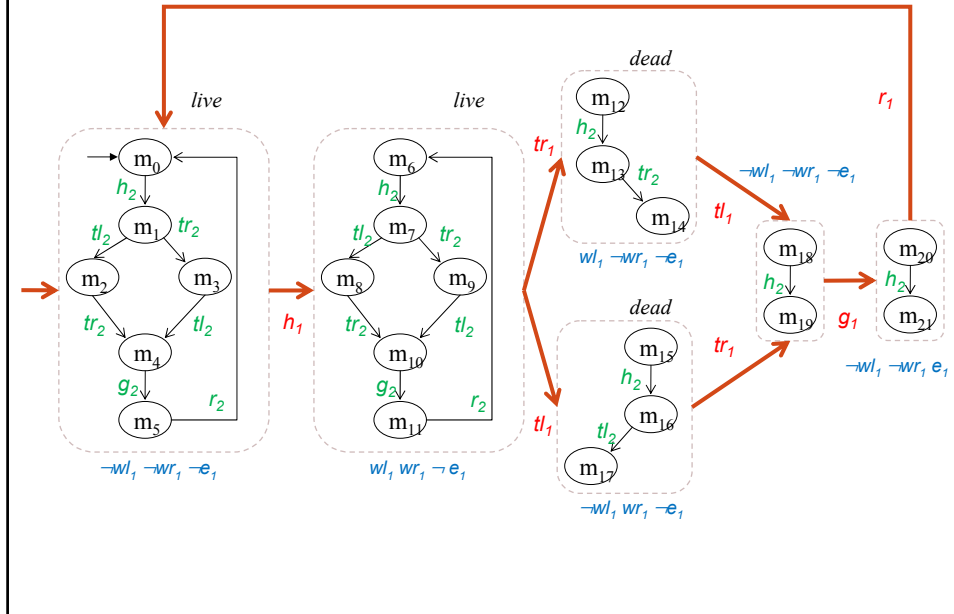
SOG construction for two philosophers



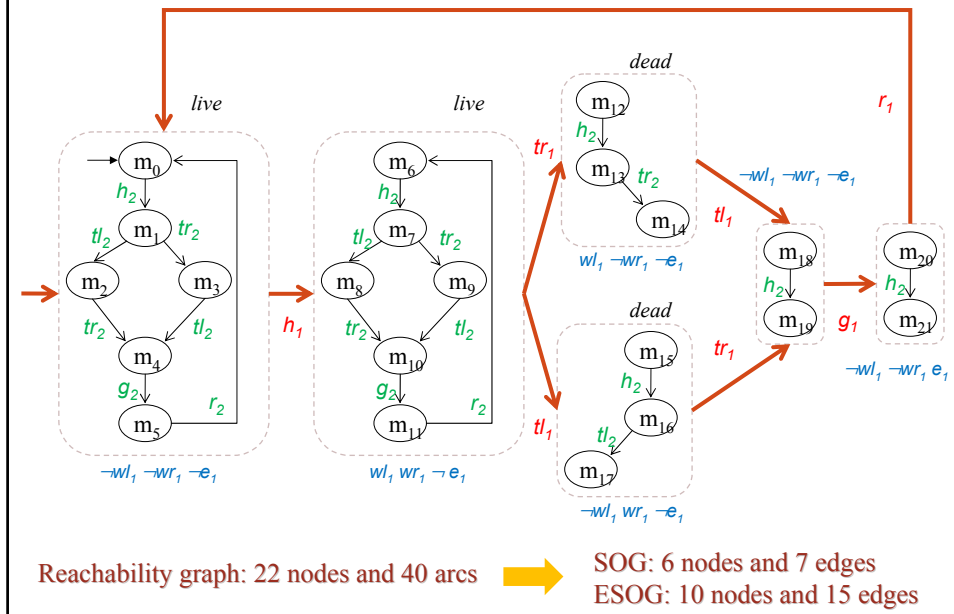
SOG construction for two philosophers



SOG construction for two philosophers



SOG construction for two philosophers



Experimental results

model				Symbolic		Symbolic & Explicit				Explicit		
		(1)	(2)	(3)	(4)	(3)	(5)	(6)	(4)	(5)	(6)	(4)
fms	3	4.8 10 ⁴	24	8.0 10 ⁵	9.2	2.3 10 ⁴	1.3 10 ³	2.0 10 ³	6.8	9.0 10 ⁴	3.4 10 ⁵	1.8
fms	4	4.3 10 ⁵	24	8.4 10 ⁵	25.7	5.1 10 ⁴	2.6 10 ³	7.0 10 ³	93.2	6.4 10 ⁵	2.6 10 ⁶	15.2
fms	5	2.8 10 ⁶	24	8.6 10 ⁵	48.6	9.1 10 ⁴	4.9 10 ³	1.5 10 ⁴	677	3.3 10 ⁶	1.4 10 ⁷	98.5
kanban4		4.5 10 ⁵	30	7.8 10 ⁵	19.1	5.1 10 ⁴	2.6 10 ³	7.1 10 ³	2.9	1.4 10 ⁶	1.1 10 ⁷	328
kanban5		2.5 10 ⁶	30	8.1 10 ⁵	47	9.4 10 ⁴	4.7 10 ³	1.4 10 ⁴	7.5	NOT	TREA	TED
kanban6		1.1 10 ⁷	30	8.2 10 ⁵	124	1.7 10 ⁵	7.9 10 ³	2.6 10 ⁴	18.9	NOT	TREA	TED
philo	6	10 ⁴	25	7.4 10 ⁵	11.6	1.8 10 ⁴	521	683	1.1	6.2 10 ⁴	2.9 10 ⁵	2.1
philo	8	2.1 10 ⁵	22	7.9 10 ⁵	32.3	2.4 10 ⁴	552	730	2.2	NOT	TREA	TED
philo	10	4.7 10 ⁶	23	8.4 10 ⁵	98.7	3.0 10 ⁴	546	724	4.0	NOT	TREA	TED
philo	20	10 ⁴	23	4.7 10 ⁶	2.5 10 ³	5.9 10 ⁴	587	916	41.5	NOT	TREA	TED
ring	3	504	43	8.2 10 ⁵	10.8	1.1 10 ⁴	1.6 10 ³	4.3 10 ³	1.3	7.5 10 ³	2.4 10 ⁴	0.2
ring	4	5.1 10 ³	40	9.1 10 ⁵	66.6	1.8 10 ⁴	1.9 10 ³	4.0 10 ³	15.2	5.9 10 ⁴	2.9 10 ⁵	1.5
ring	5	5.3 10 ⁴	39	9.8 10 ⁵	438	6.9 10 ⁴	9.6 10 ³	2.9 10 ⁴	612	8.0 10 ⁵	4.9 10 ⁶	181

- (1) Number of reachable markings (4) Verification time in seconds
 (2) Number of verified formulae (5) Number of visited states
 (3) Peak number of live BDD nodes (6) Number of visited transitions

Heuristics

- The bad results obtained for the “fms” and “ring” models are essentially due to the computation time of aggregates
- The SOG definition offers some freedom degrees for the construction
- In particular, an aggregate can be split into several ones (introducing some stuttering)
- In our application to Petri nets, we have experimented the tool when some supplementary transitions (arbitrary chosen) are considered as observed

Experimental results

model				Symbolic		Symbolic & Explicit				Explicit		
		(1)	(2)	(3)	(4)	(3)	(5)	(6)	(4)	(5)	(6)	(4)
fms	5	$2.8 \cdot 10^6$	24	$8.6 \cdot 10^5$	48.6	$9.1 \cdot 10^4$	$1.1 \cdot 10^4$	$2.6 \cdot 10^4$	52.1	$3.3 \cdot 10^6$	$1.4 \cdot 10^7$	98.5
fms	6	$1.5 \cdot 10^7$	24	$8.8 \cdot 10^5$	147.6	$2.9 \cdot 10^5$	$1.7 \cdot 10^4$	$4.5 \cdot 10^4$	182	NOT	TREA	TED
ring	5	$5.3 \cdot 10^4$	39	$9.8 \cdot 10^5$	438	$1.0 \cdot 10^5$	$5.6 \cdot 10^4$	$3.3 \cdot 10^5$	239	$8.0 \cdot 10^5$	$4.9 \cdot 10^6$	181

- | | | | |
|-----|-------------------------------|-----|-------------------------------|
| (1) | Number of reachable markings | (4) | Verification time in seconds |
| (2) | Number of verified formulae | (5) | Number of visited states |
| (3) | Peak number of live BDD nodes | (6) | Number of visited transitions |

Conclusion and perspectives

- SOG takes advantage of symbolic and explicit approaches
 - Symbolic state spaces
 - On-the-fly model checking (only the needed part of the graph is constructed)
- Adaptation to other decision diagrams (DDD, SDD)
- Adaptation to modular systems
- Combination with partial order approaches
- Dynamic computation of the atomic propositions which must be taken into account
- ...