A Counterexample-Based Incremental and Modular Verification Approach

Étienne André Kais Klai Hanen Ochi Laure Petrucci

LIPN, University Paris 13
Villetaneuse, France

Motivation

Aims

Verification of LTL properties φ

Usual approach

- Convert the negation of φ into a Büchi automaton
- Synchronise the Büchi automaton and the system model
- Check that the composed system language is empty

Drawback

State space explosion problem

Motivation

Aims

Verification of LTL properties φ

Usual approach

- Convert the negation of φ into a Büchi automaton
- Synchronise the Büchi automaton and the system model
- Check that the composed system language is empty

Drawback

State space explosion problem

Motivation

Incremental Model-Checking [Klai Petrucci Reniers FORTE07]

- Decomposition of a Petri net according to the formula to check
- Abstraction of the environment
- Construction of local counterexamples
- Incremental processing until the property is proven true or false

Motivation

Symbolic Observation Graph [Haddad Ilié Klai ATVA04]

- Reduced graph
- Dedicated to the verification of LTL\X properties

Modular approach [Klai Petrucci ACSD08]

- Smaller structures
- Efficient especially with loosely coupled modules
- Allows for reuse
- Modular construction of the Symbolic Observation Graph for the verification of LTL\X properties

Motivation

Symbolic Observation Graph [Haddad Ilié Klai ATVA04]

- Reduced graph
- Dedicated to the verification of LTL\X properties

Modular approach [Klai Petrucci ACSD08]

- Smaller structures
- Efficient especially with loosely coupled modules
- Allows for reuse
- Modular construction of the Symbolic Observation Graph for the verification of LTL\X properties

Motivation

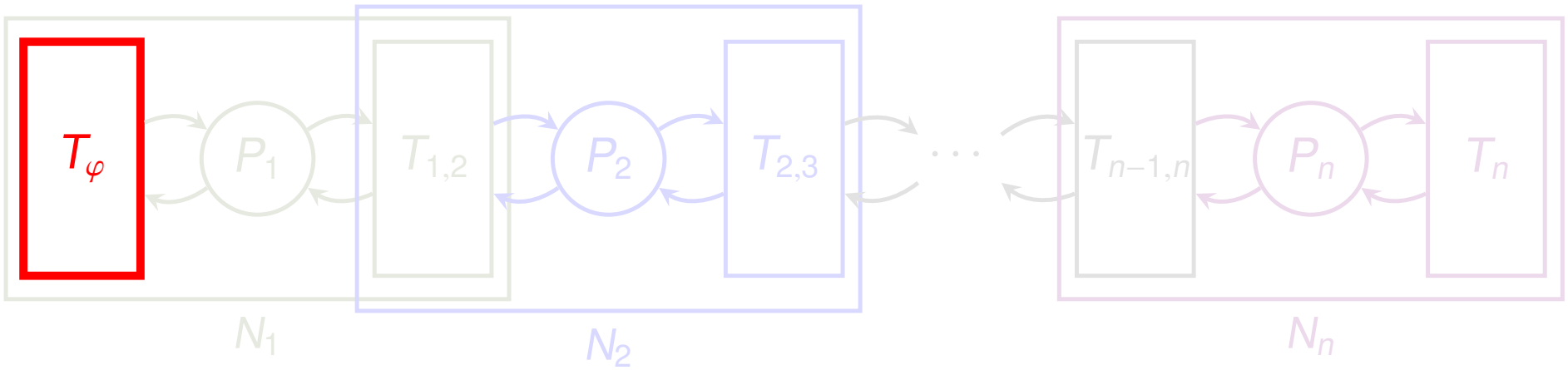
$\Rightarrow$ Combination of both approaches

Outline

1. Incremental Model-Checking [Klai Petrucci Reniers FORTE07]
   - Model decomposition
   - Abstraction of the environment
   - Incremental verification
2. Modular SOG construction [Klai Petrucci ACSD08]
   - Symbolic Observation Graph
   - Modular construction
3. Incremental and modular counterexample search
   - Drawbacks of previous approaches
   - Optimisation of the SOG Construction
   - Incremental verification revisited
4. Conclusion & Perspectives
   - Summary
   - Perspectives for future work

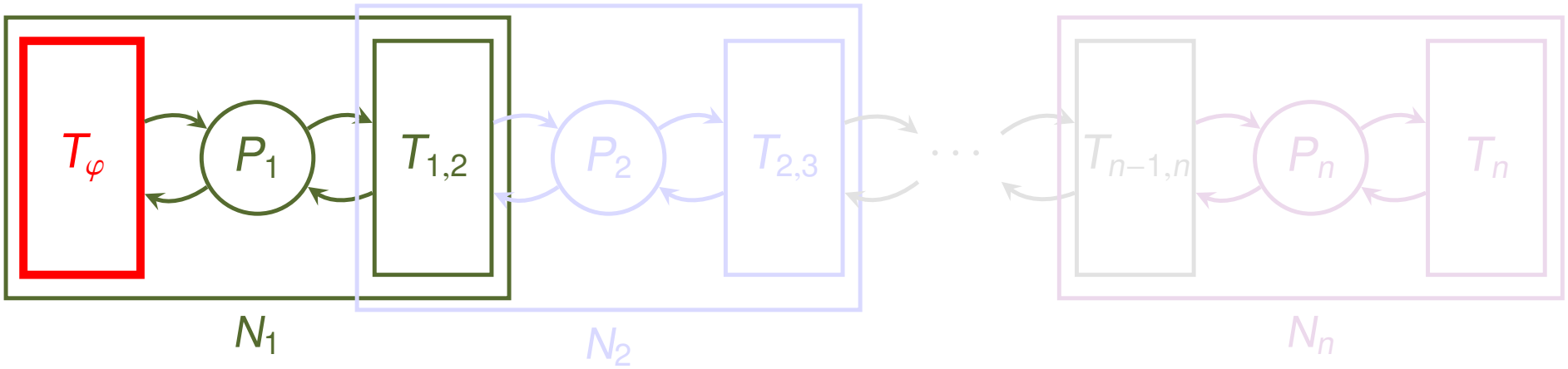
Model decomposition

Petri net models with fused transitions



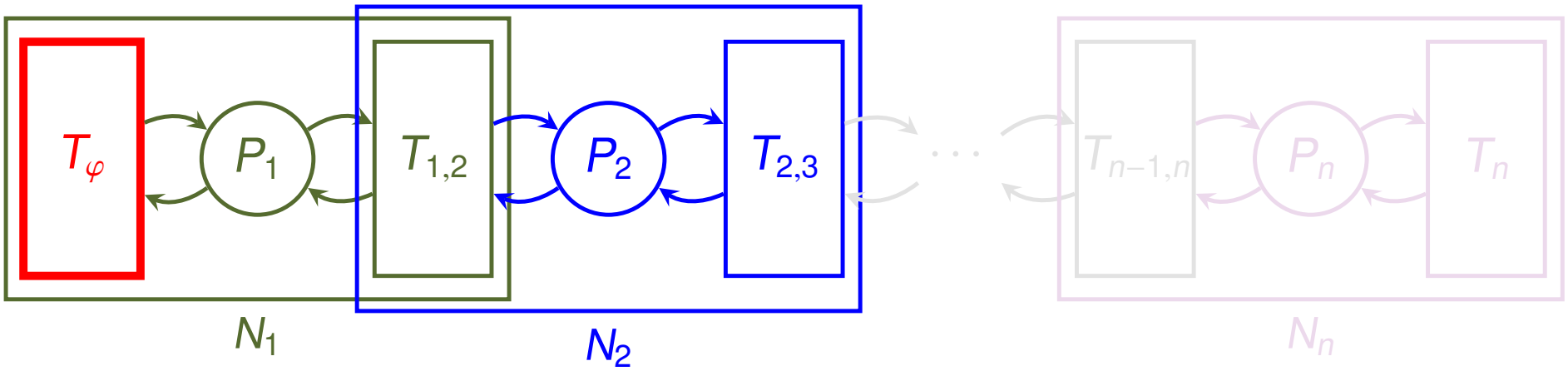
Model decomposition

Petri net models with fused transitions



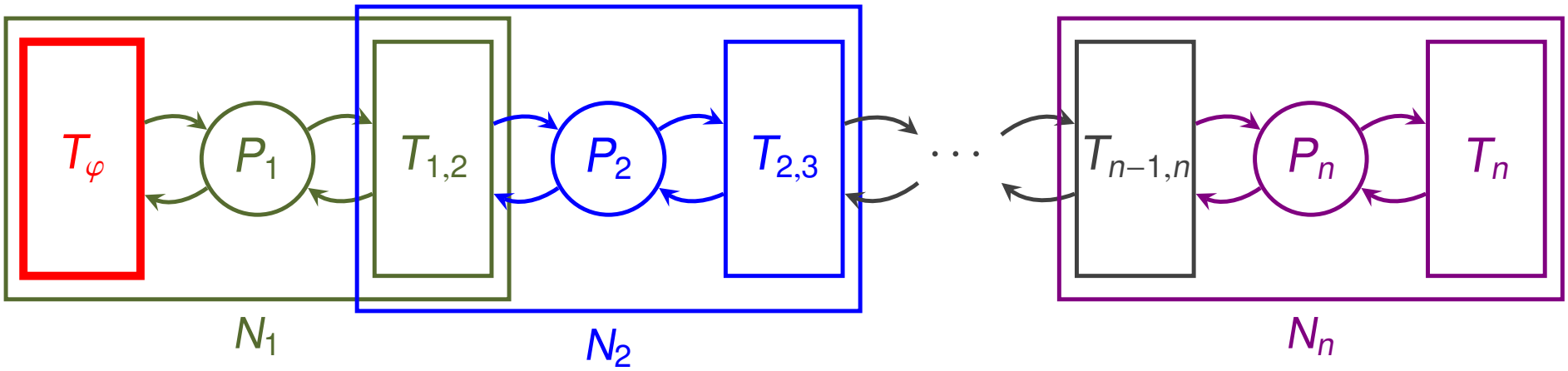
Model decomposition

Petri net models with fused transitions



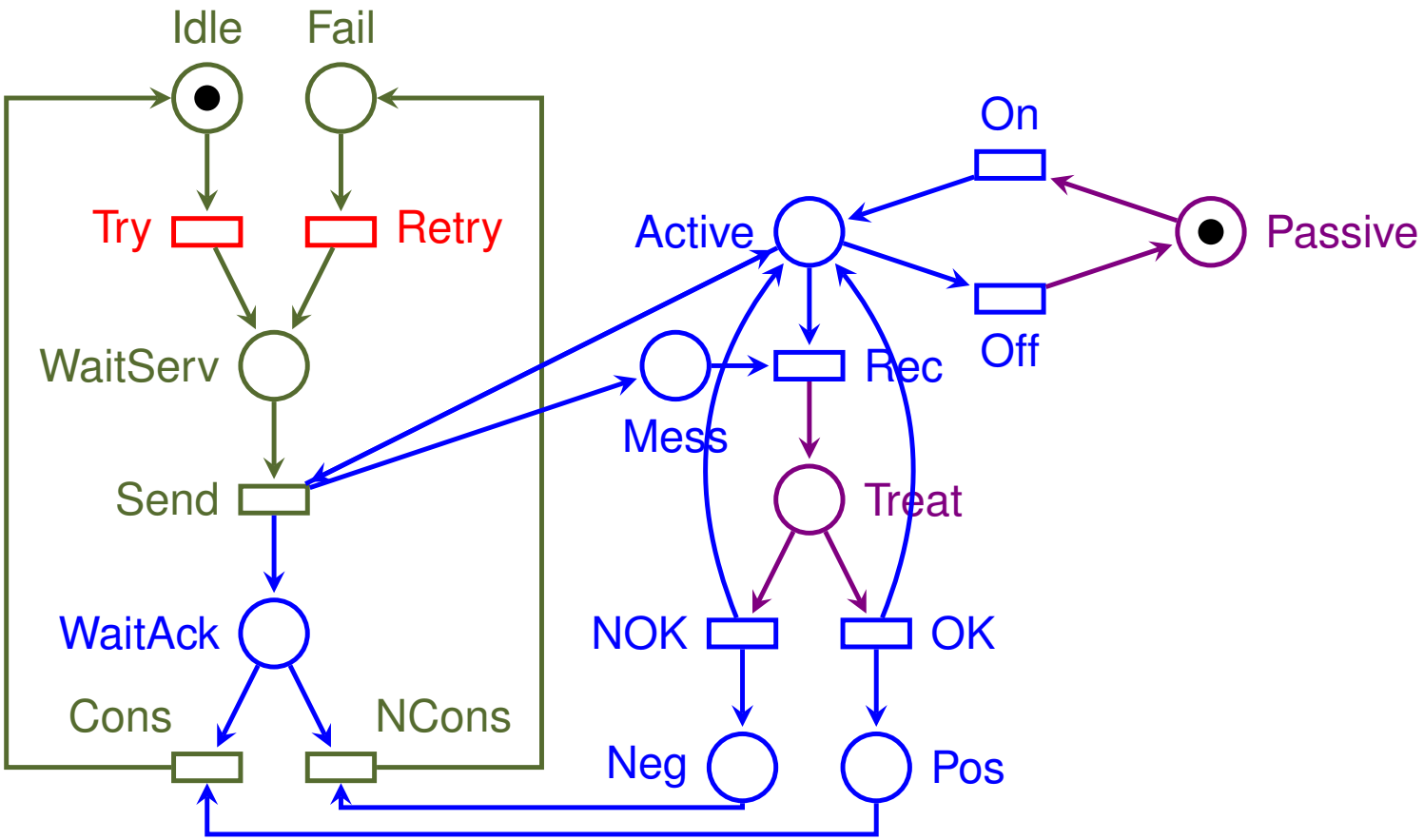
Model decomposition

Petri net models with fused transitions



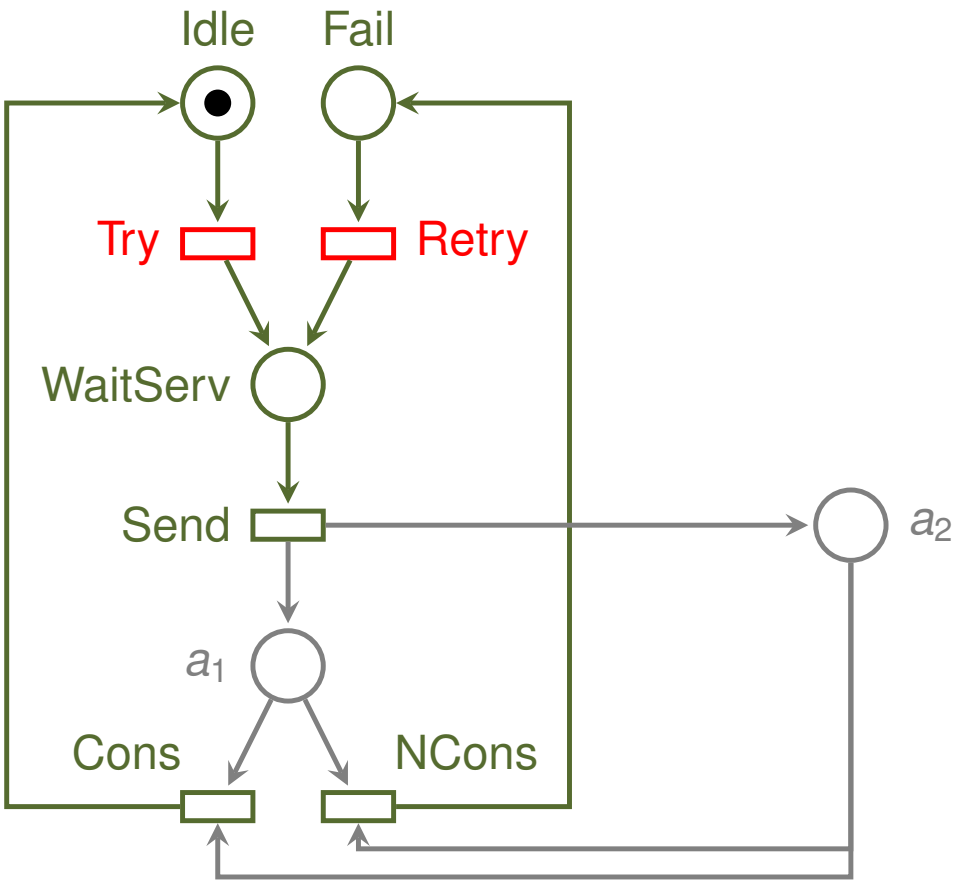
Abstracting the environment

- Control the interface using additional places
- Abstract the interface with one place per global invariant

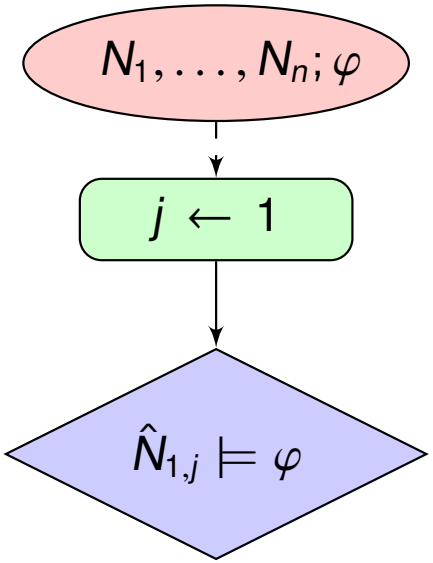


Abstracting the environment

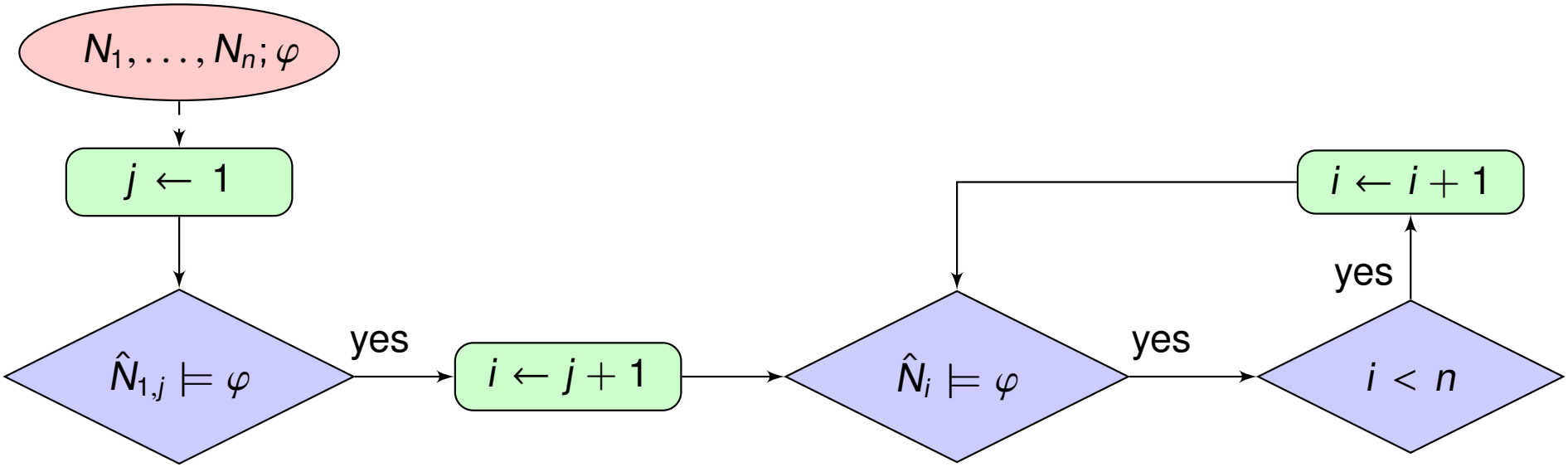
- Control the interface using additional places
- Abstract the interface with one place per global invariant



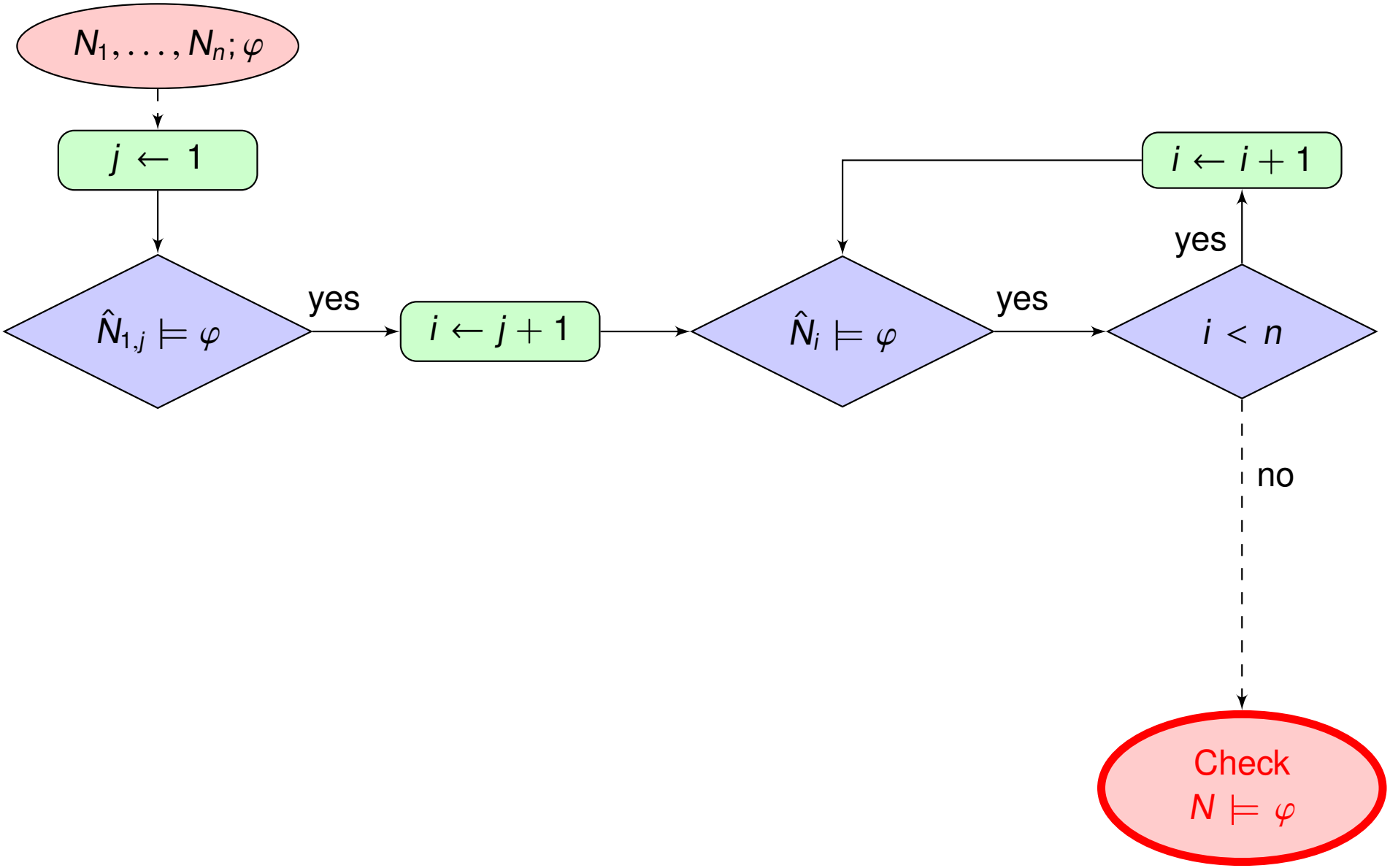
Incremental verification



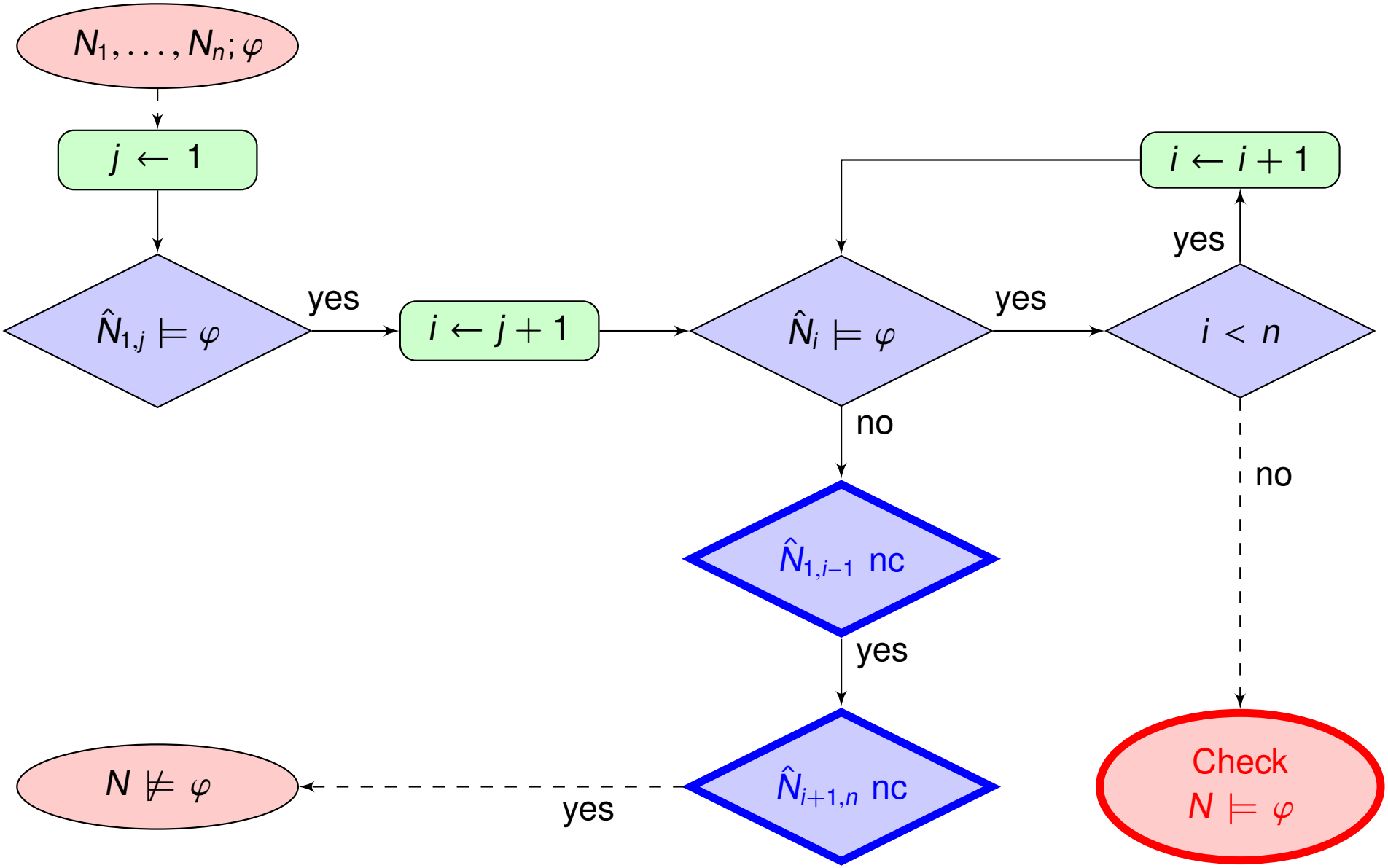
Incremental verification



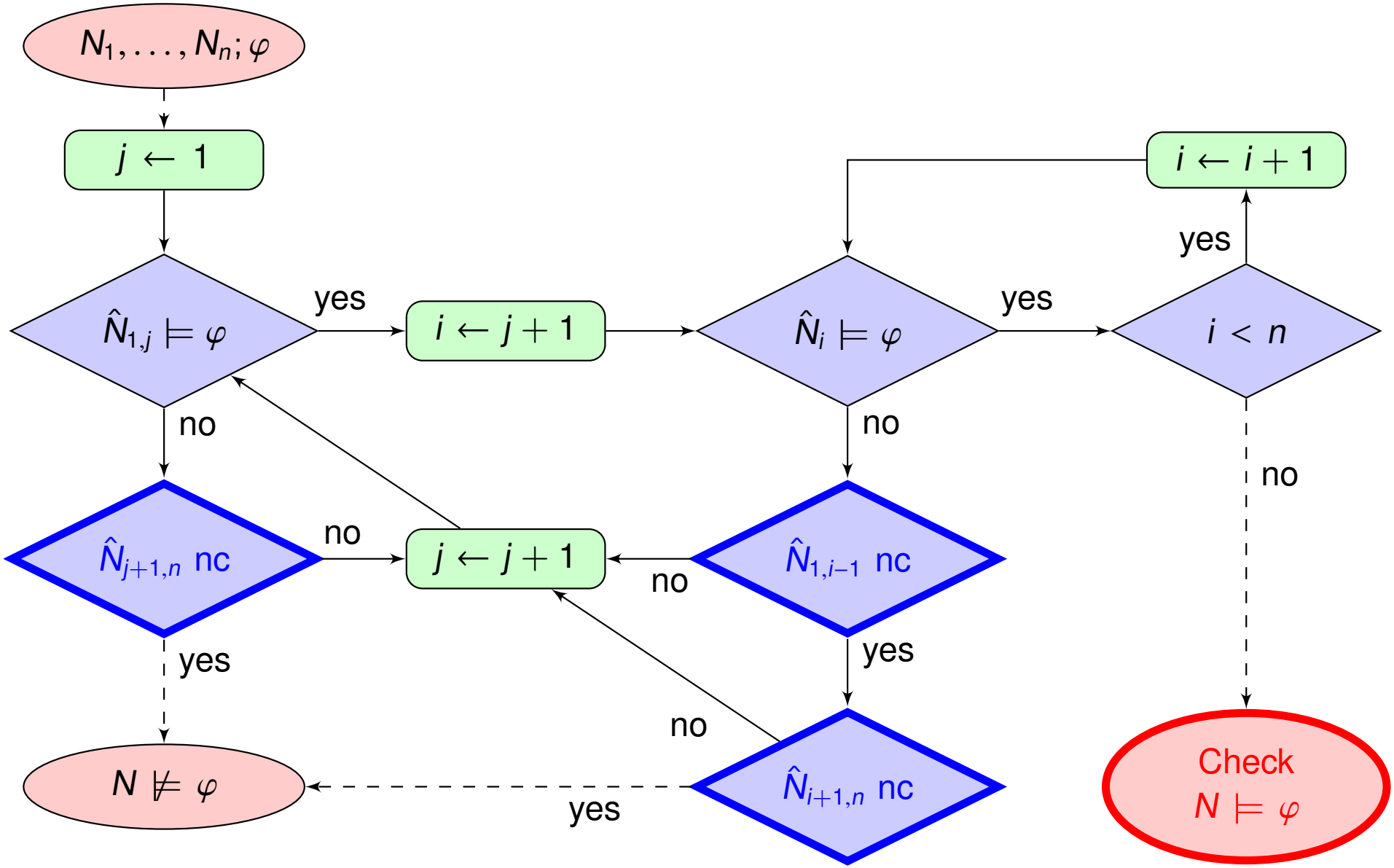
Incremental verification



Incremental verification



Incremental verification



Symbolic Observation Graph

Observable/Unobservable actions

Partition the set of actions: $Act = UnObs \cup Obs$ where:

- *Obs* contains the observable actions, occurring in the formula φ
- *UnObs* contains the unobservable actions, i.e. all the other ones

Symbolic Observation Graph

Observable/Unobservable actions

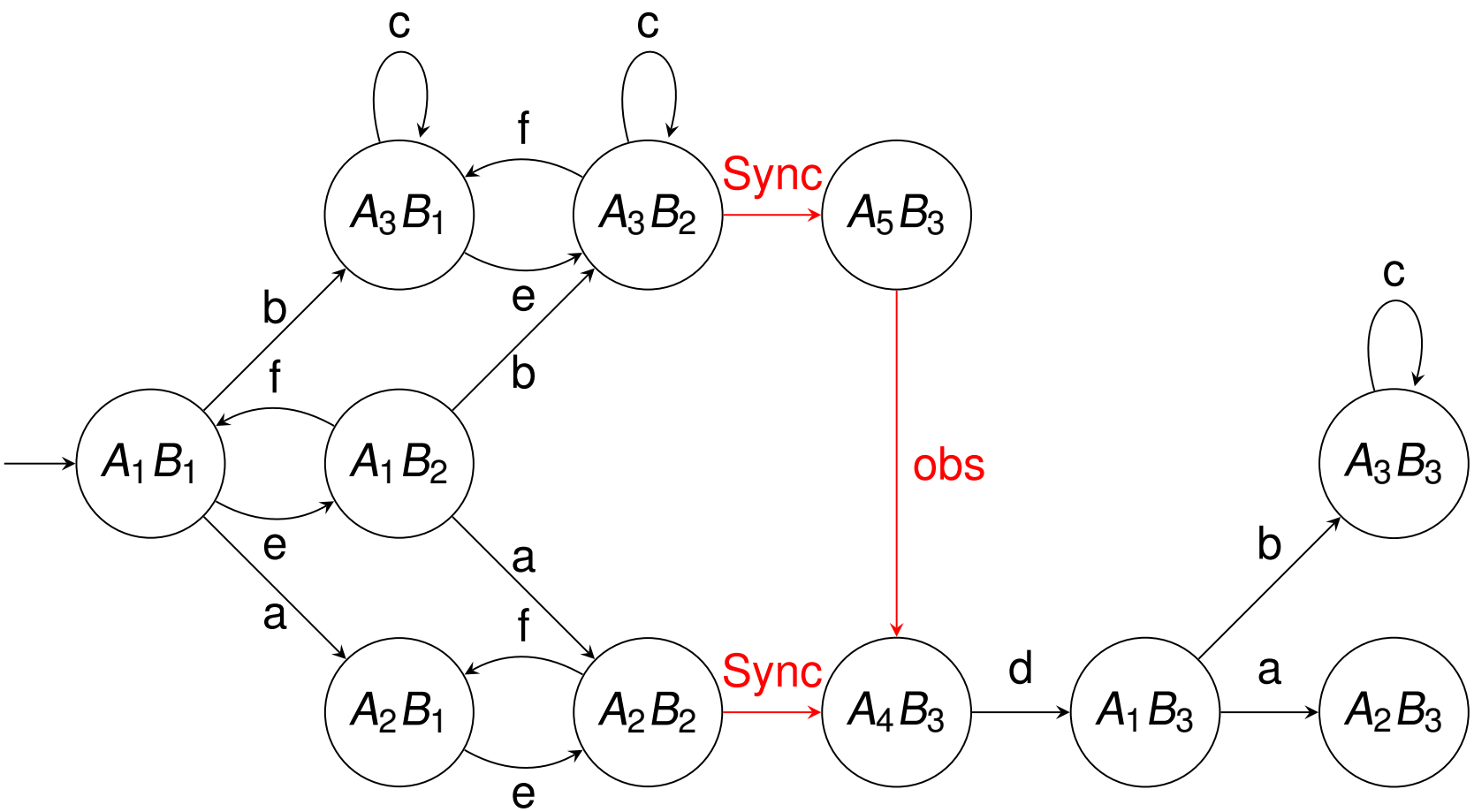
Partition the set of actions: $Act = UnObs \cup Obs$ where:

- *Obs* contains the observable actions, occurring in the formula φ
- *UnObs* contains the unobservable actions, i.e. all the other ones

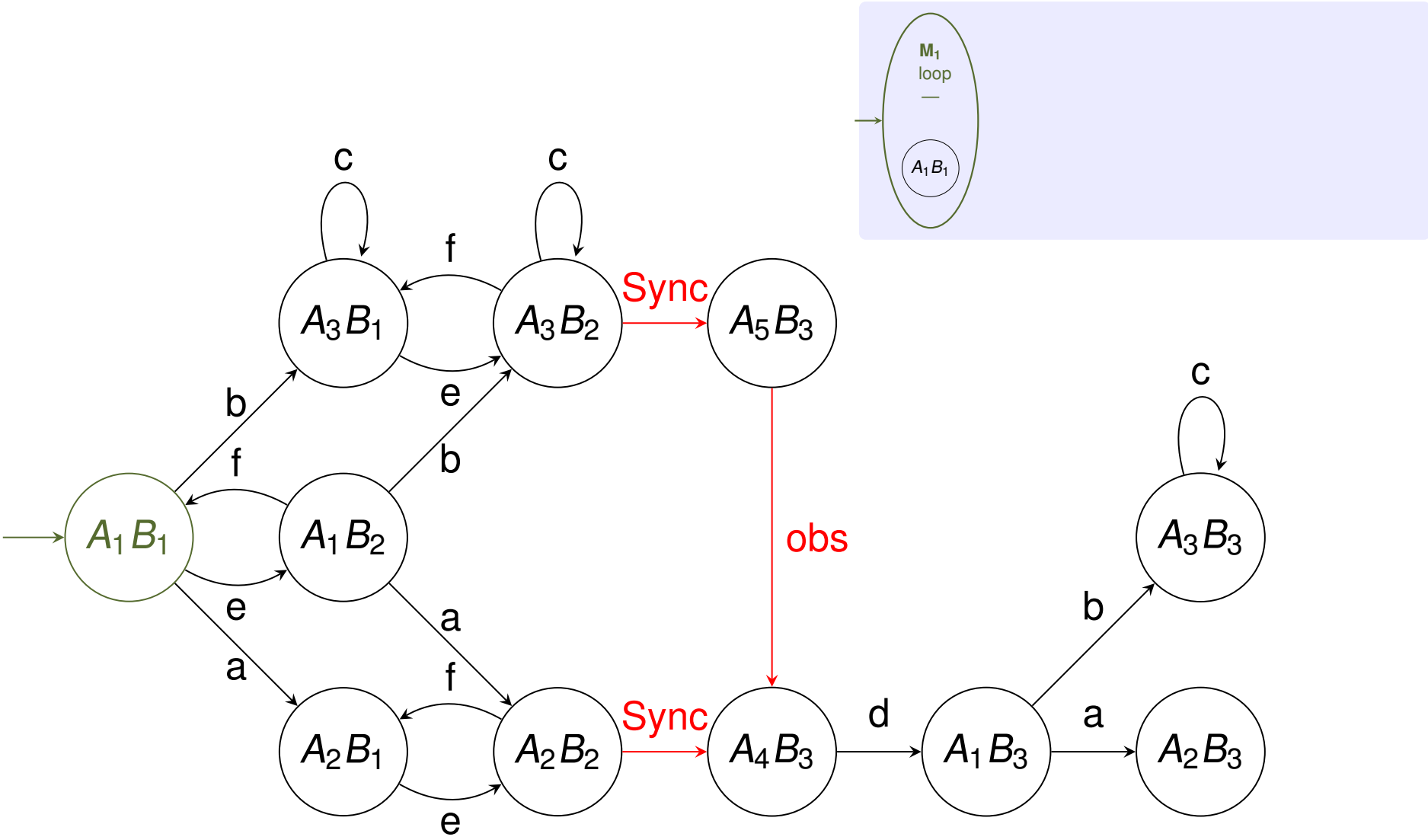
Symbolic Observation Graph

- Nodes are meta-states:
  - set of states, possibly reachable from one another using unobservable transitions
  - $loop \in \mathbb{B}$ indicating whether the meta-state contains a cycle
  - $deadlock \in \mathbb{B}$ indicating whether the meta-state contains a deadlock
- Arcs are labelled by actions in *Obs*

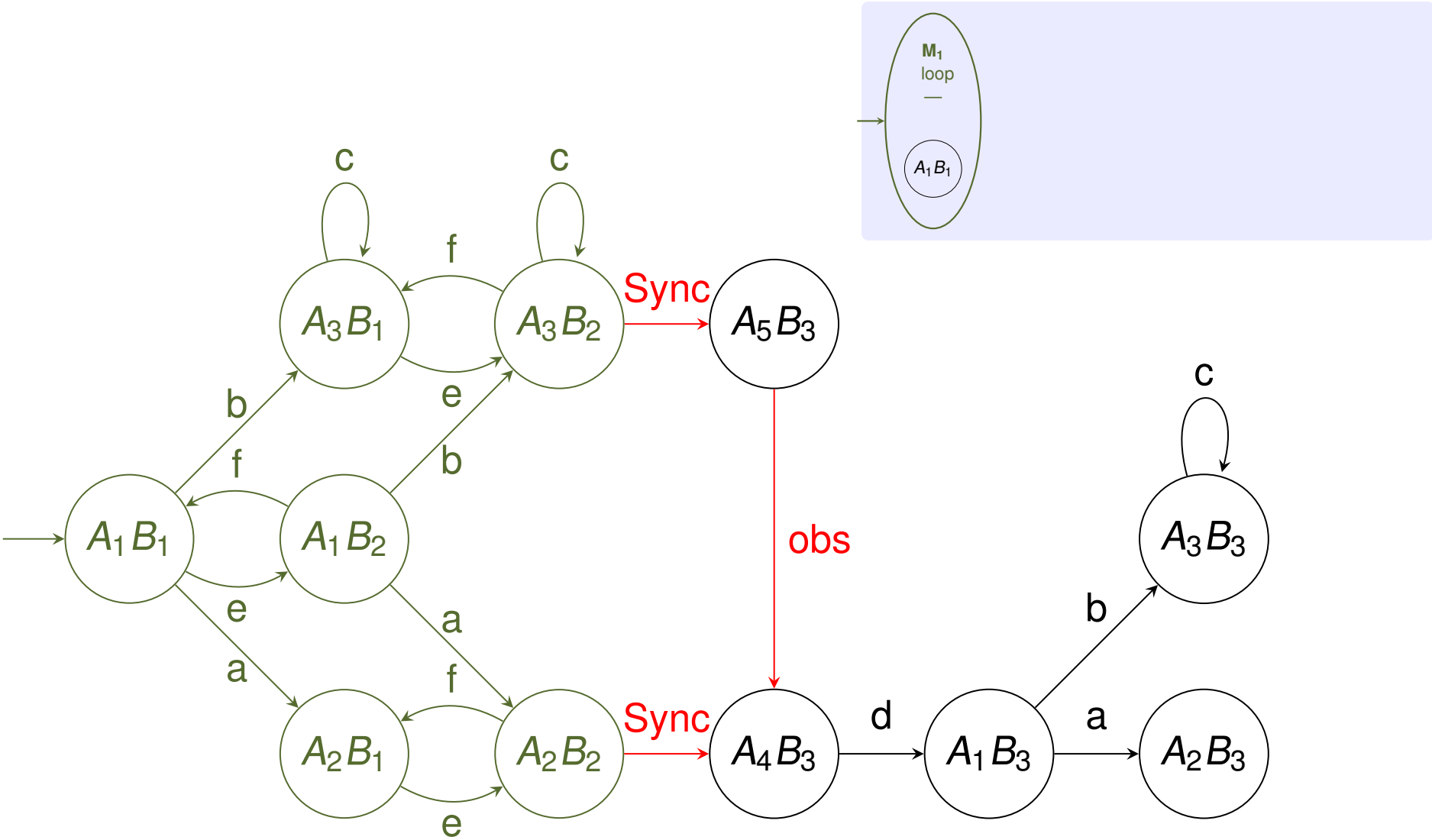
SOG Construction



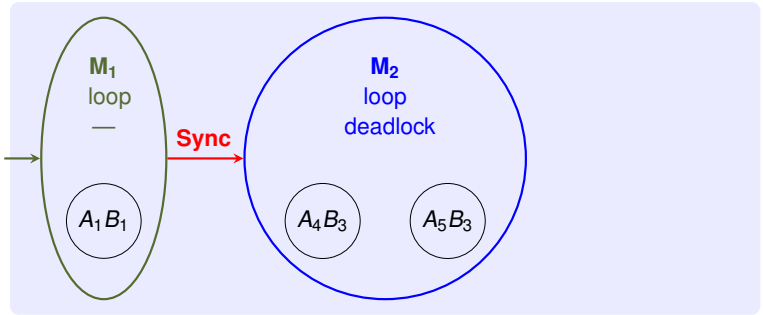
SOG Construction

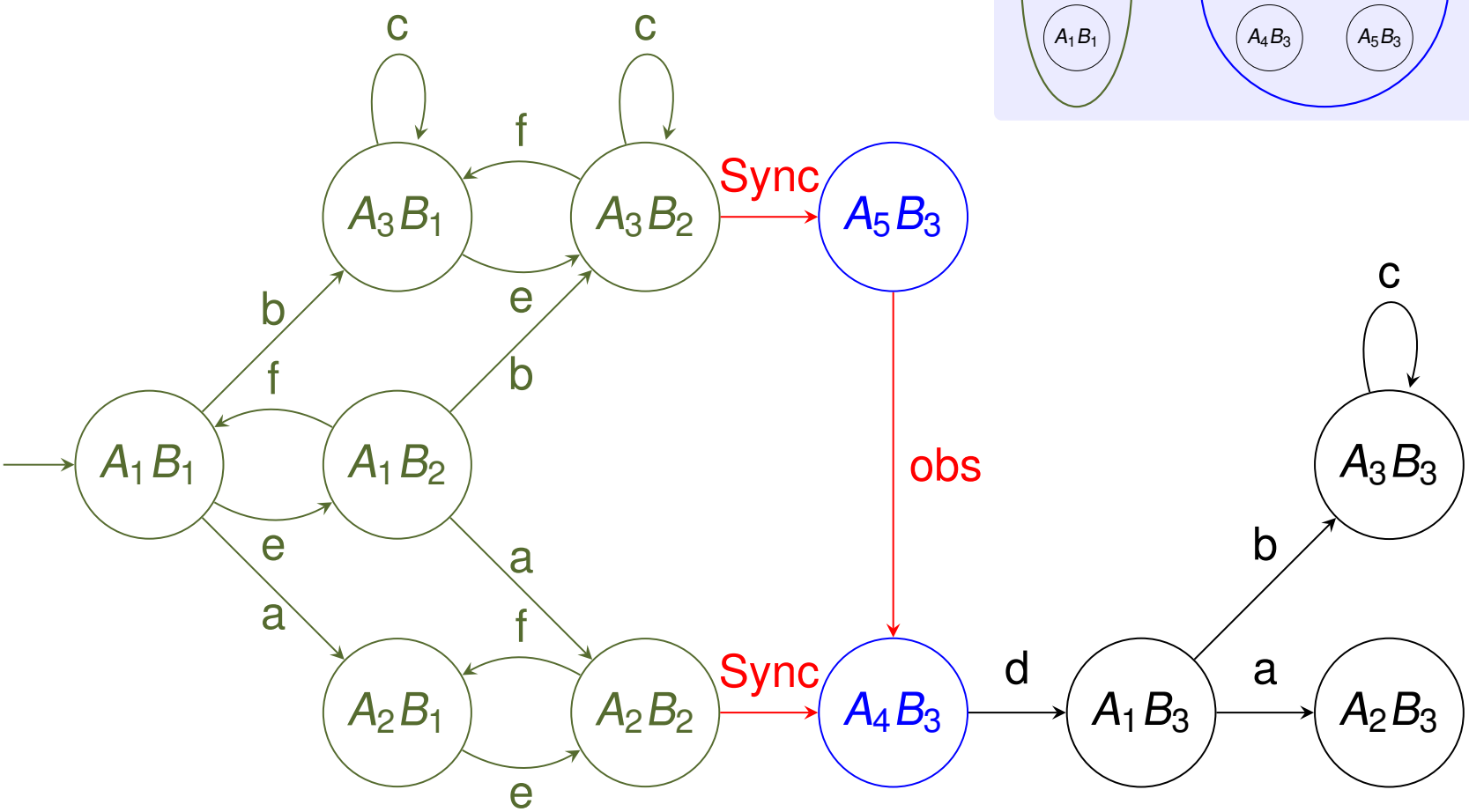


SOG Construction

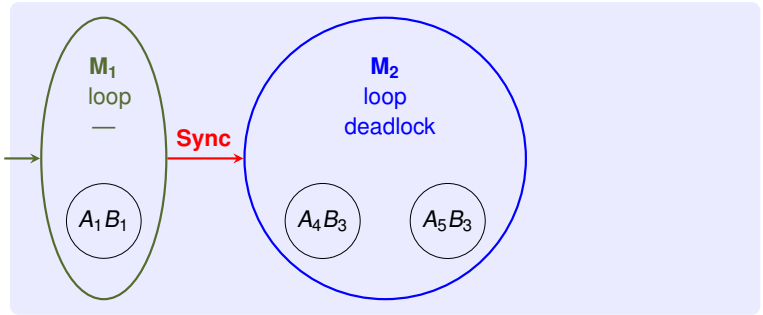


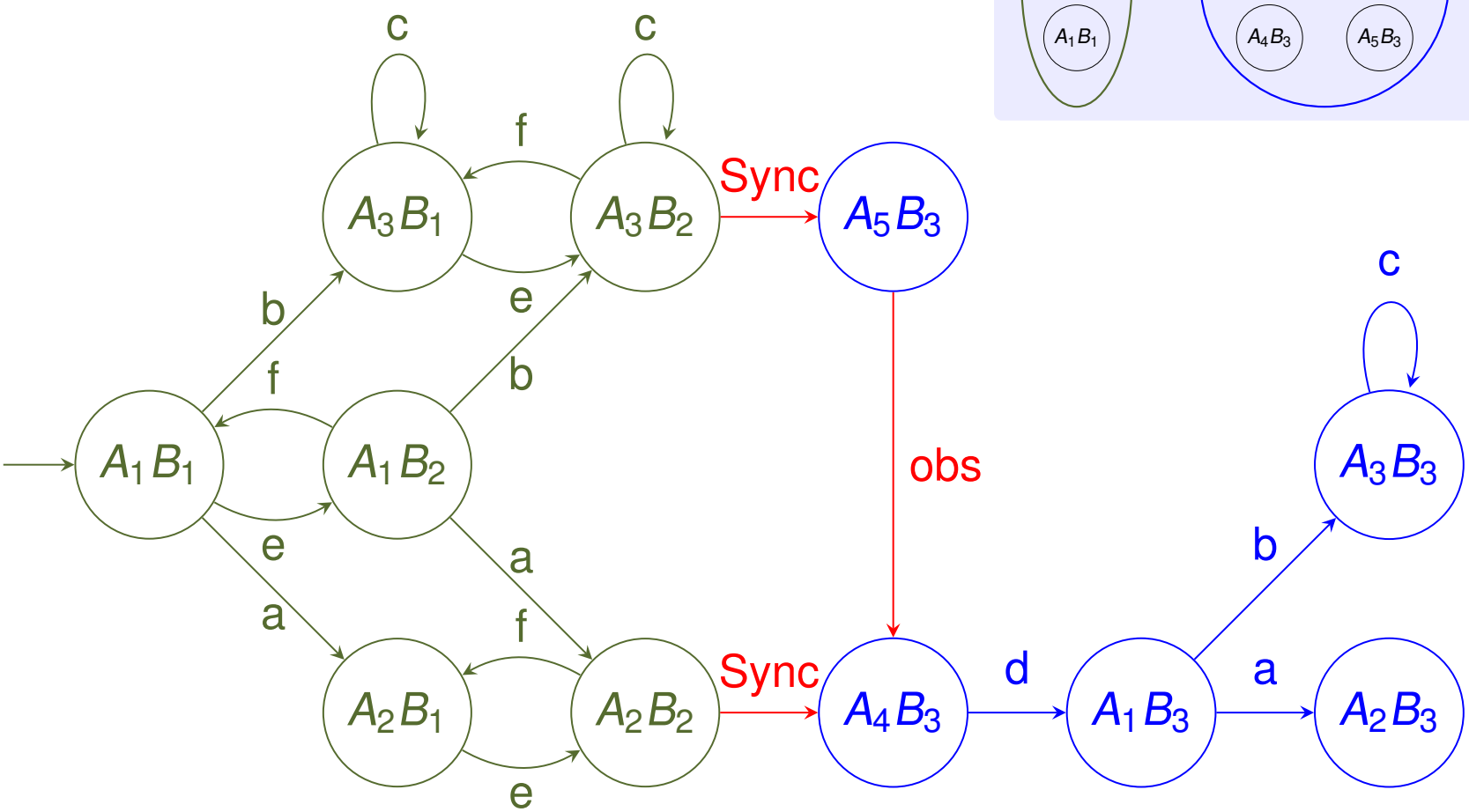
SOG Construction



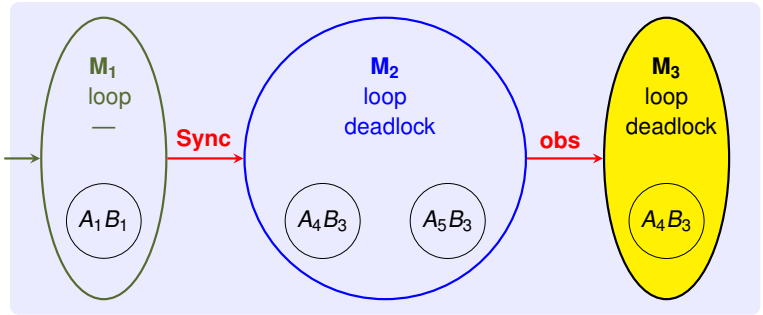


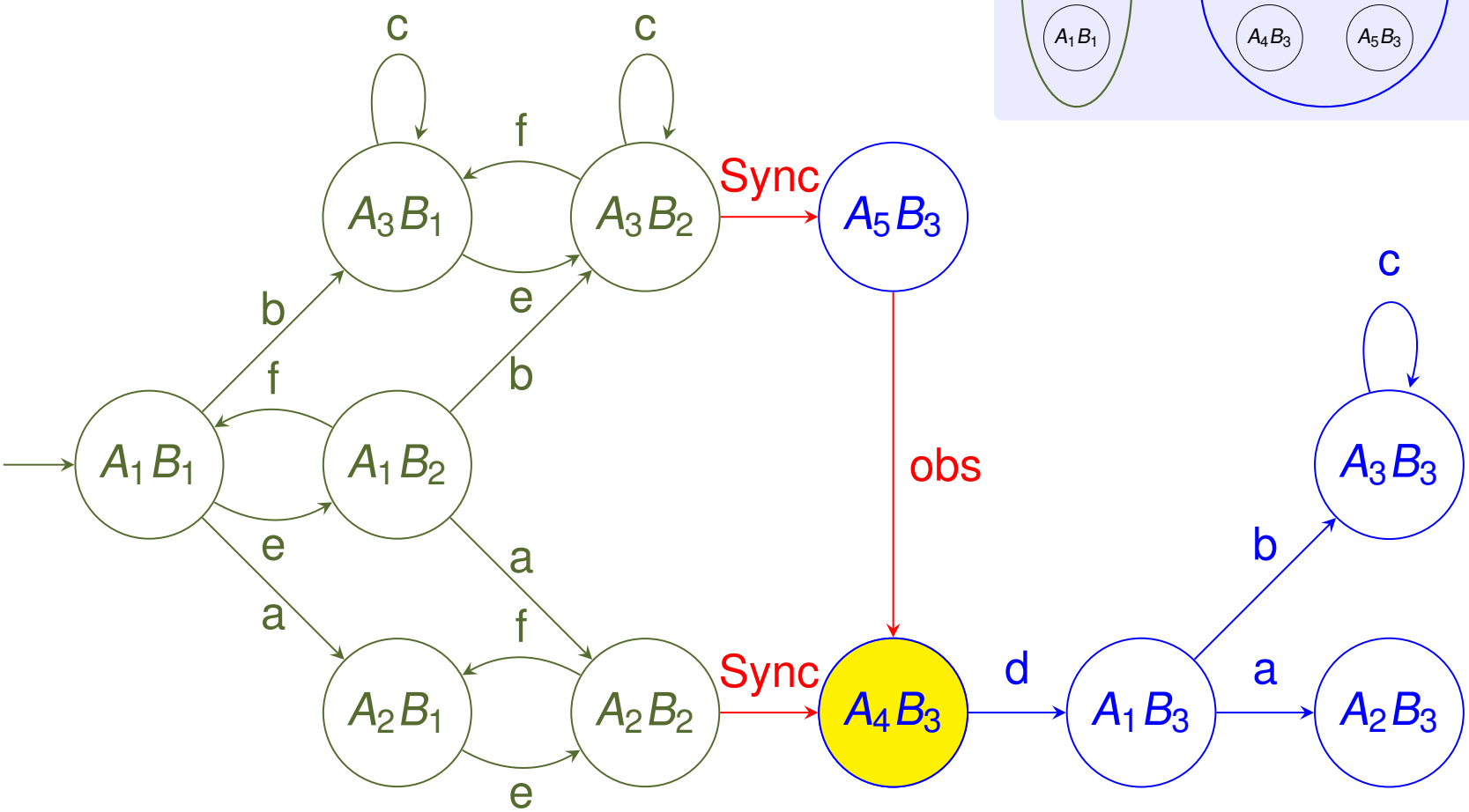
SOG Construction



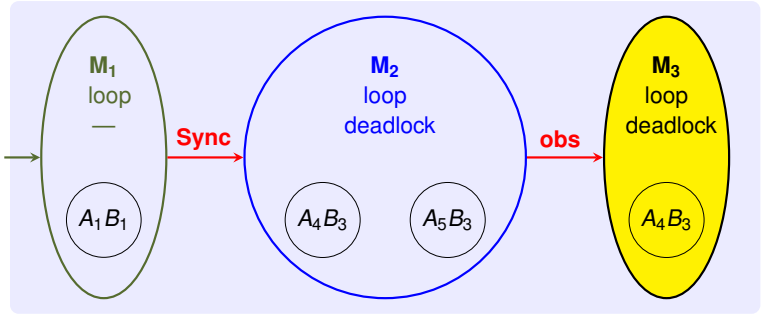


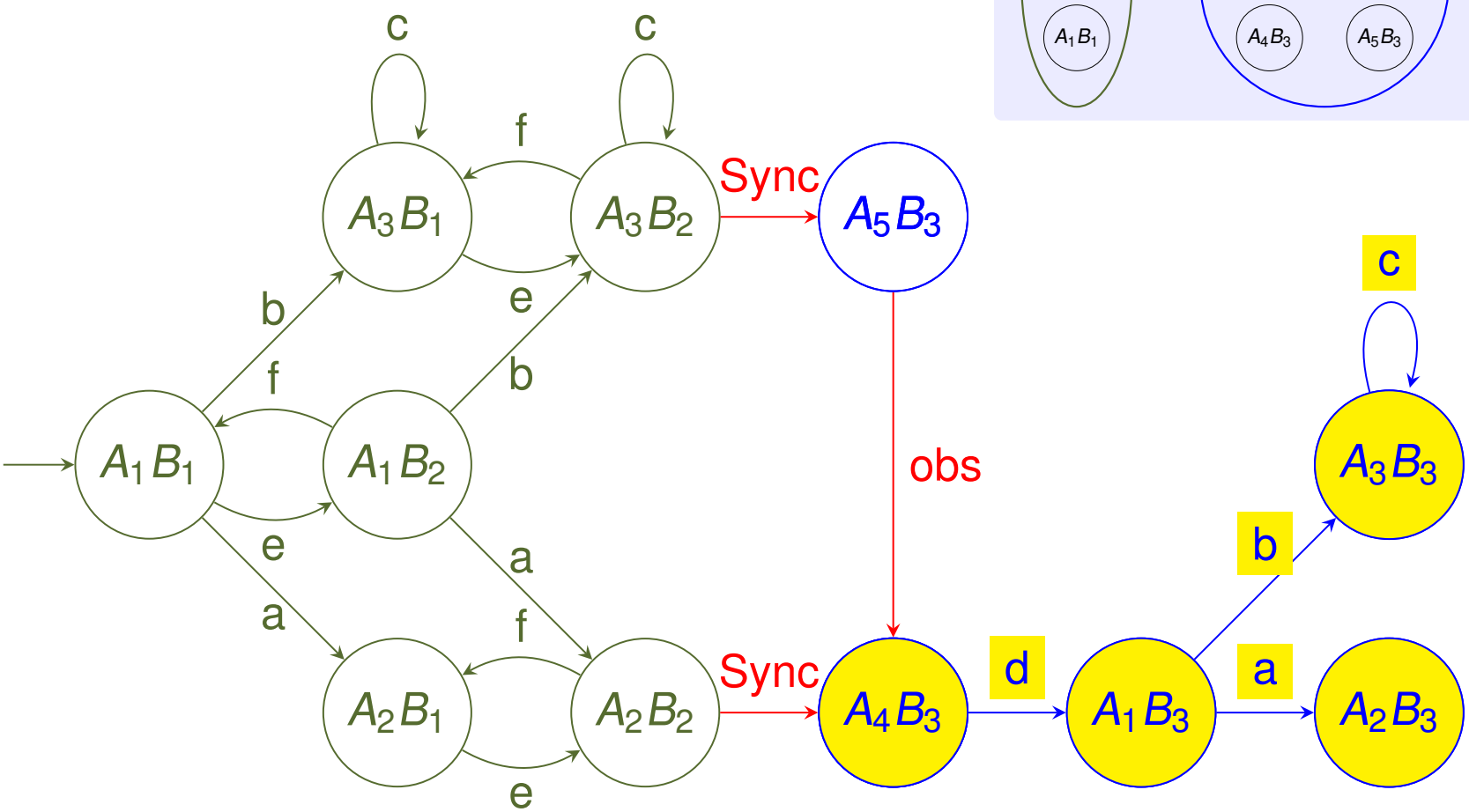
SOG Construction



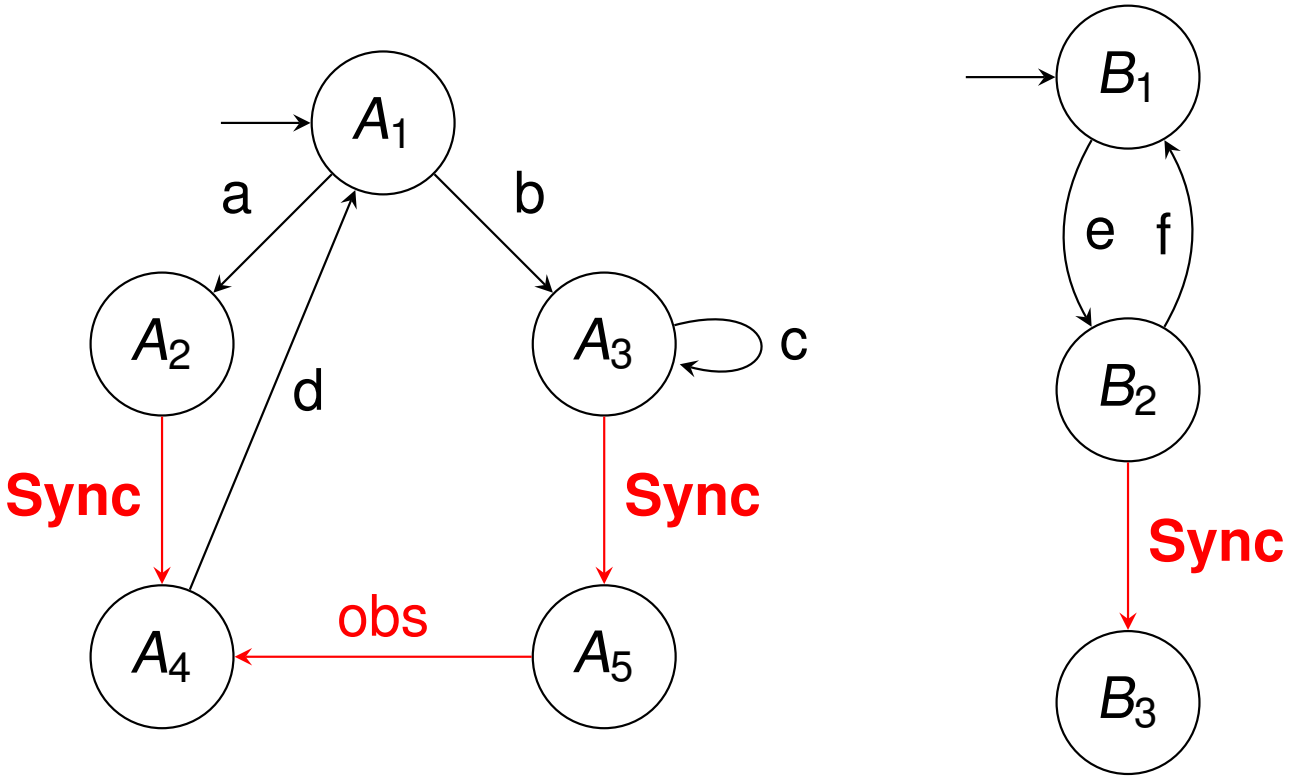


SOG Construction

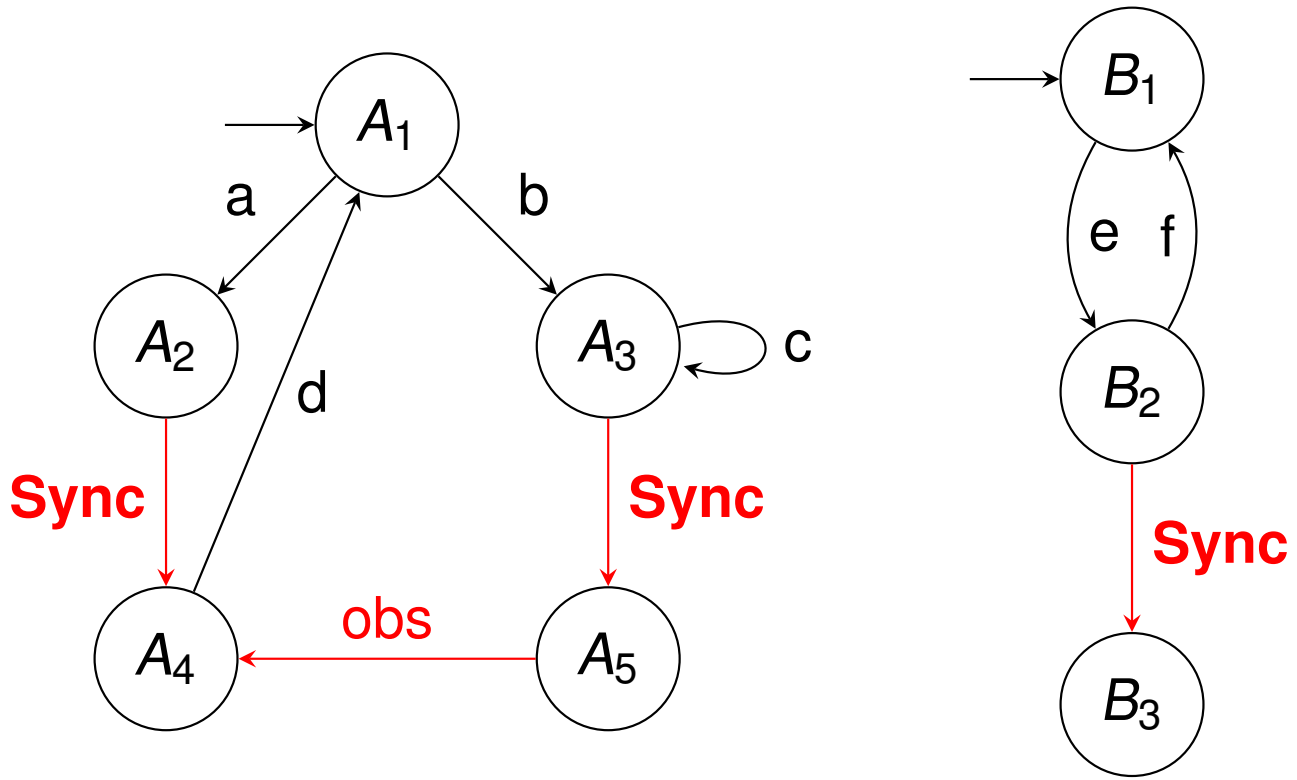




Modular Construction

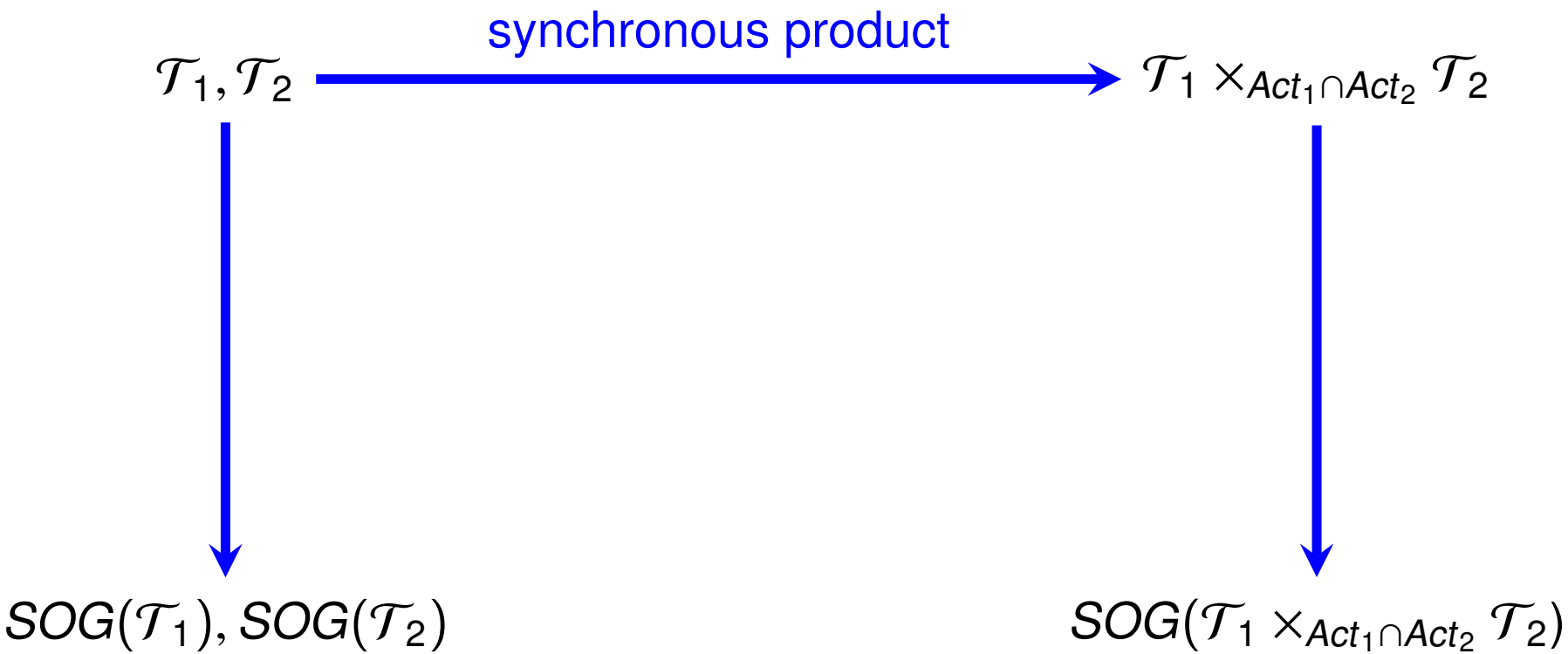


Modular Construction

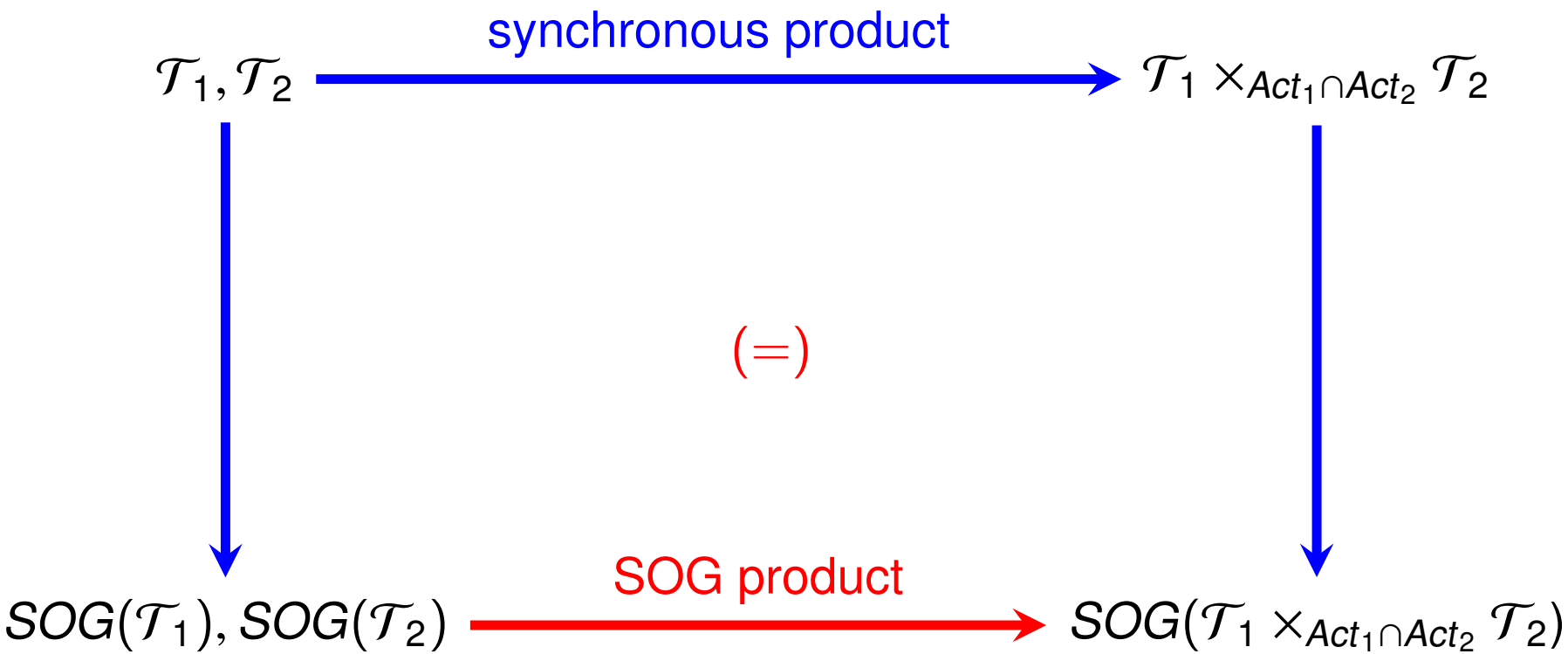


Observe both synchronised actions and those of the formula.

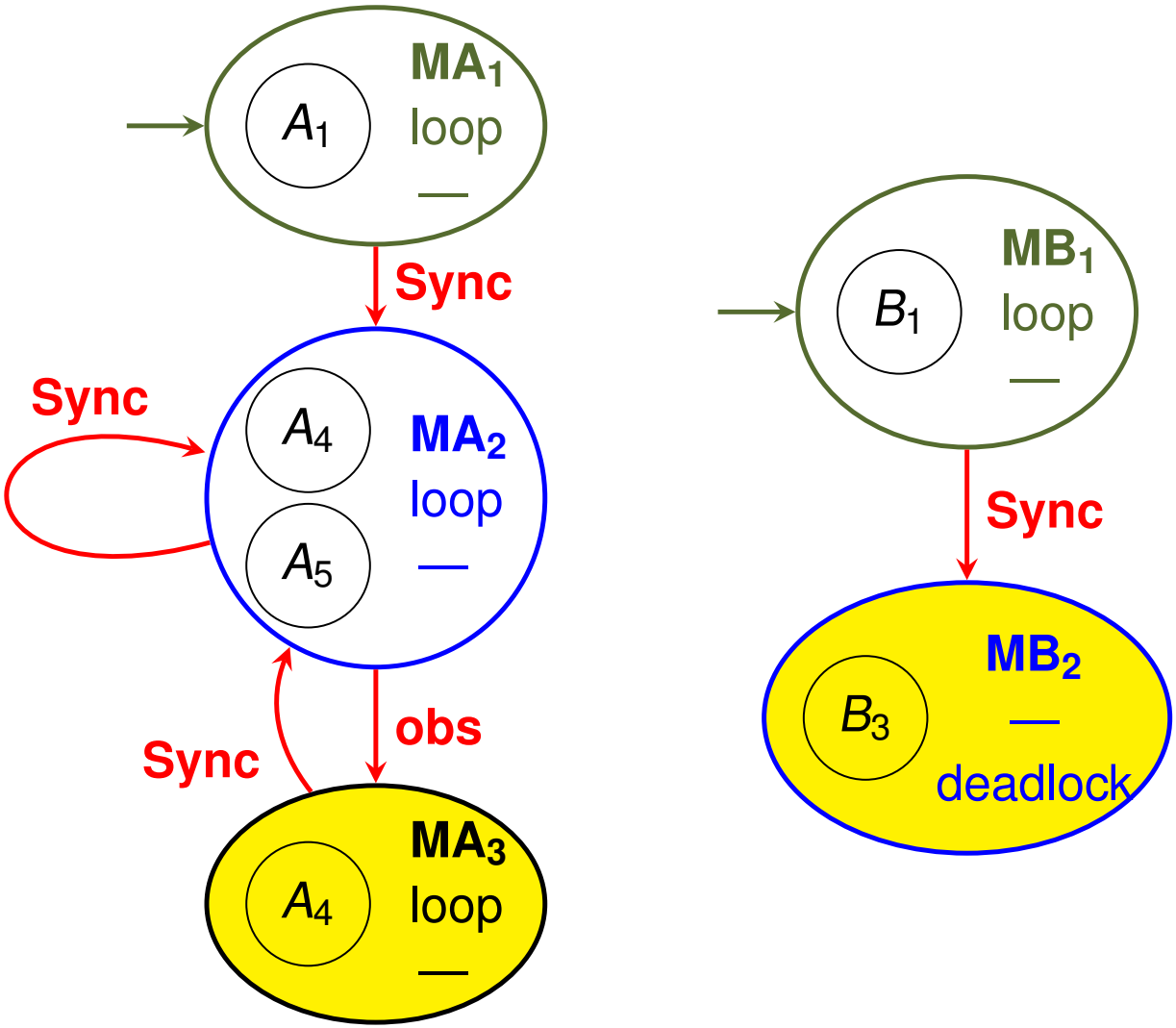
Symbolic Observation Graphs Product



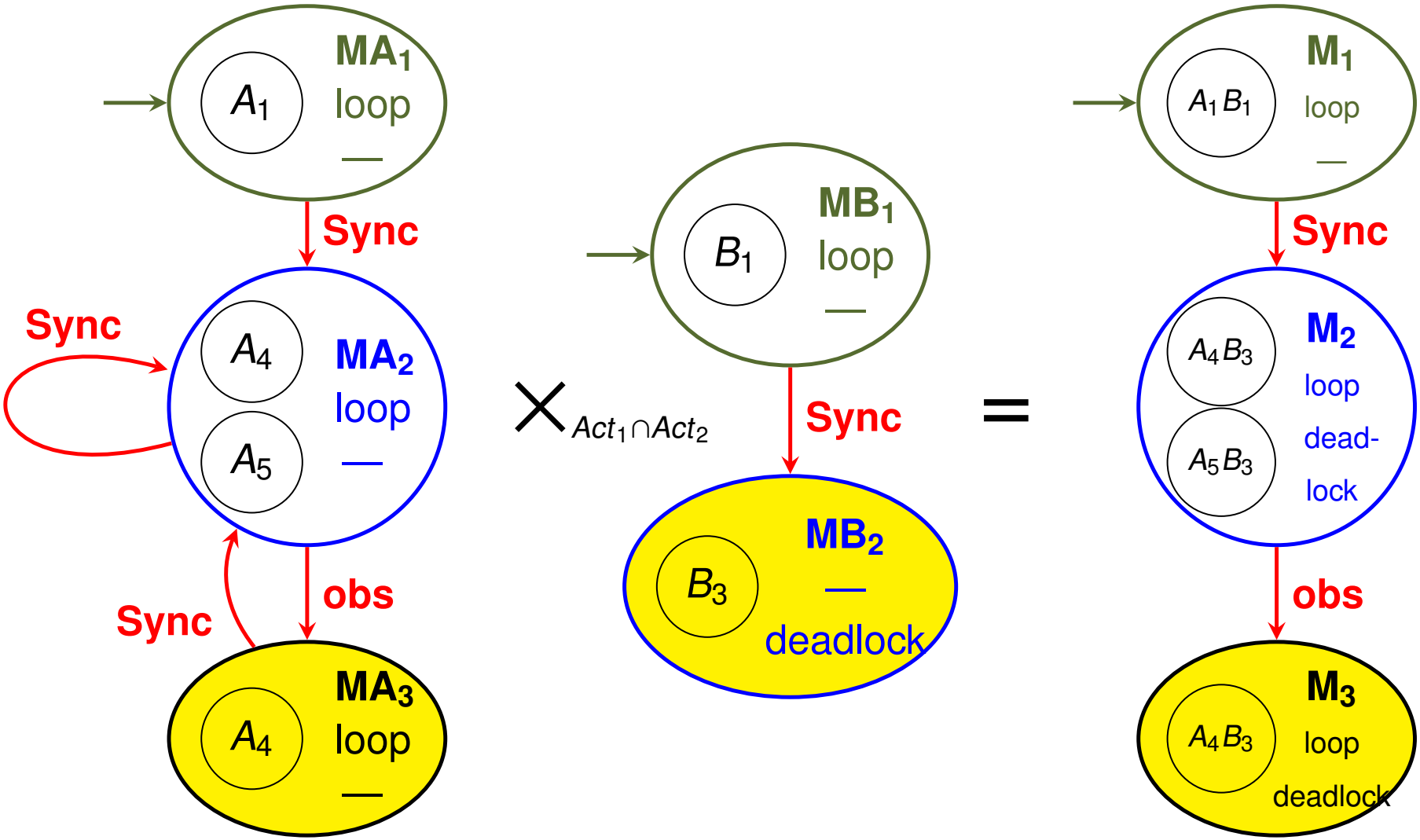
Symbolic Observation Graphs Product



Symbolic Observation Graphs Product



Symbolic Observation Graphs Product



Experimental Results

Model	**param**	N_{OG}	N_{MSS}	N_{MSOG}
5 AGVs		$30,965,760$	900	12
Database	n	$n \times 3^{n-1} + 1$	$6n + 3$	$n + 1$
Philosophers	2	3	11	3
	3	4	16	4
	n	$N_{OG}(n-1)$ $+N_{OG}(n-2)$	$N_{OG}(n) + 4n$	$N_{OG}(n)$
Poisoned philosophers	2	21	33	17
	3	99	99	75
	n	$4N_{OG}(n-1)$ $+3N_{OG}(n-2)$ $+6$	$4N_{MSS}(n-1)$ $+N_{MSS}(n-2)$ $+8n+4$	$4N_{MSOG}(n-1)$ $+N_{MSOG}(n-2)$ $+4$
Railway	n	$4(n^2+n+1)$	$\frac{n(n+1)}{2} + 5n + 10$	$\frac{n(n+1)}{2} + 2n + 3$

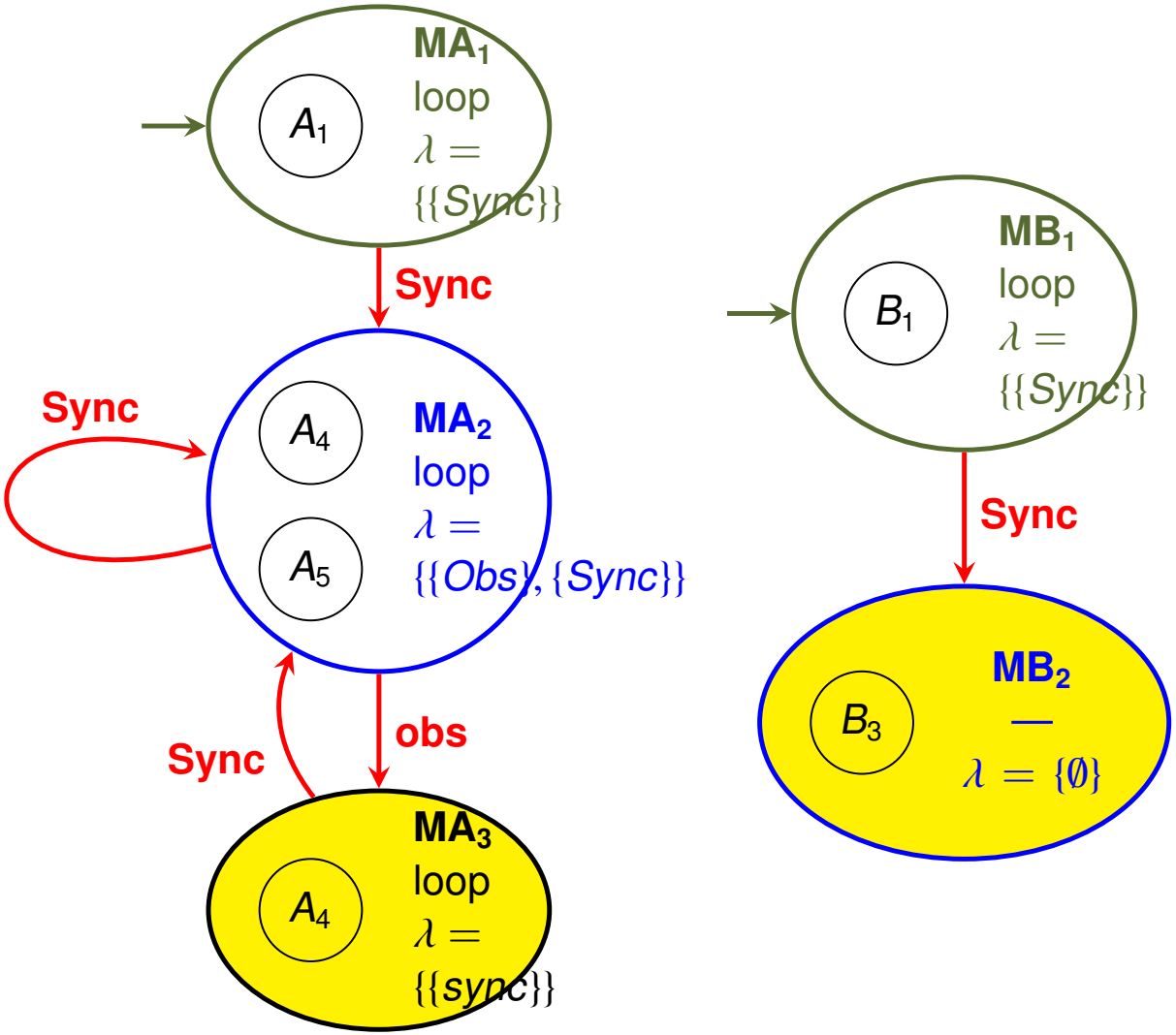
Drawbacks of Previous Approaches

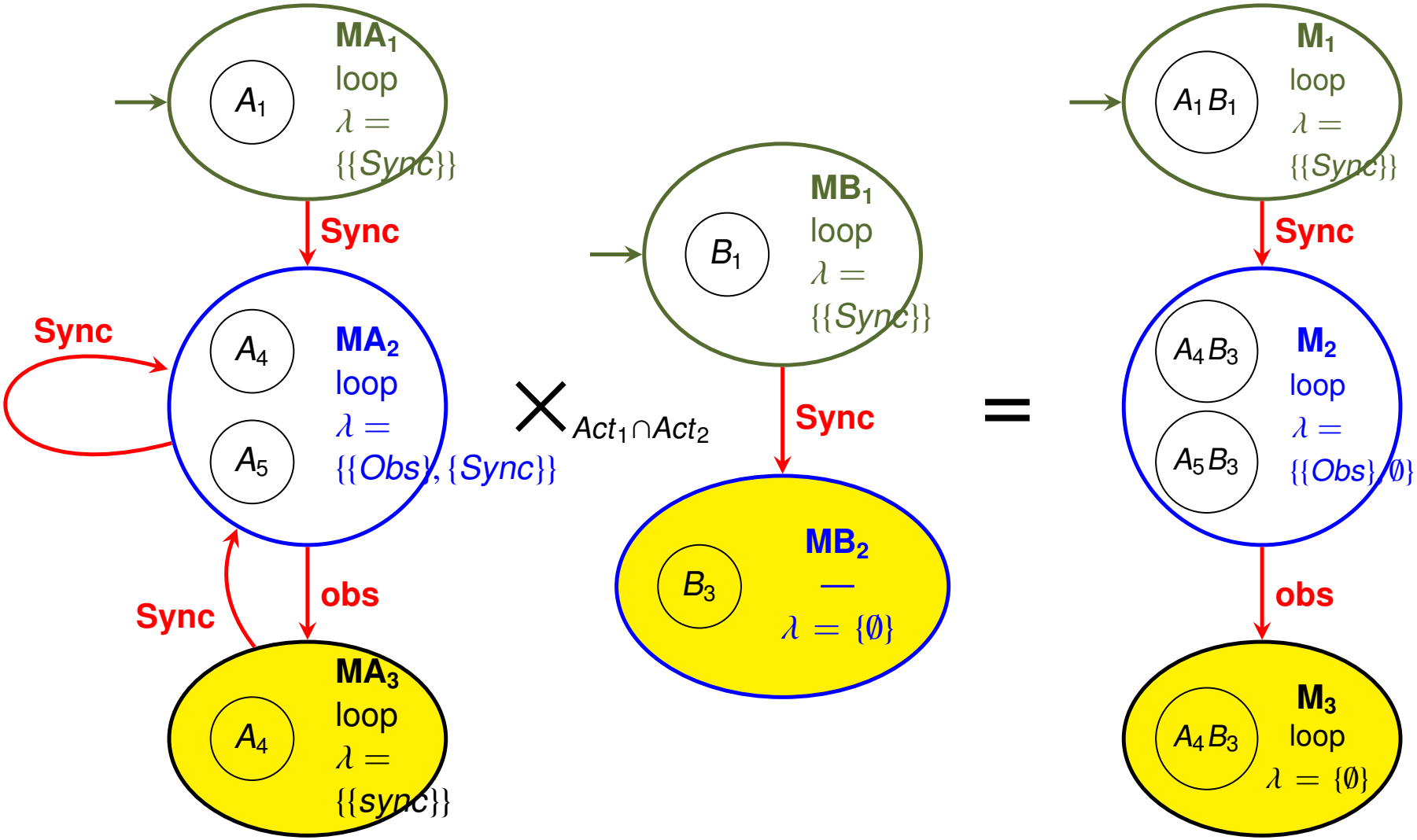
Drawbacks

1. Decomposition of the Petri net model according to the formula
2. Non-constraining relation checked using language inclusion
3. Model-checking the complete model in case the formula holds
4. Computing the synchronised meta-states deadlock attribute

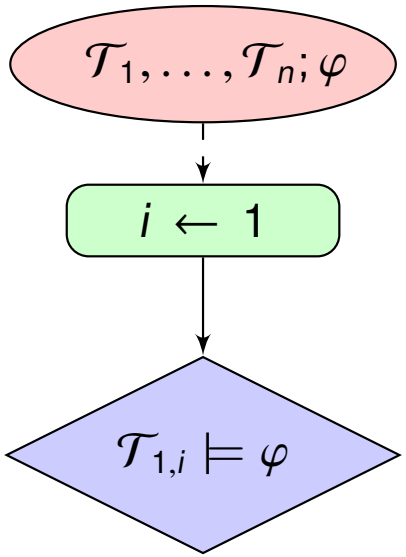
Improvements

1. Synchronised LTSs
2. Checking validity of counterexamples
3. Use of the SOG as a reduced structure for model-checking
4. Introducing an observed behaviour mapping

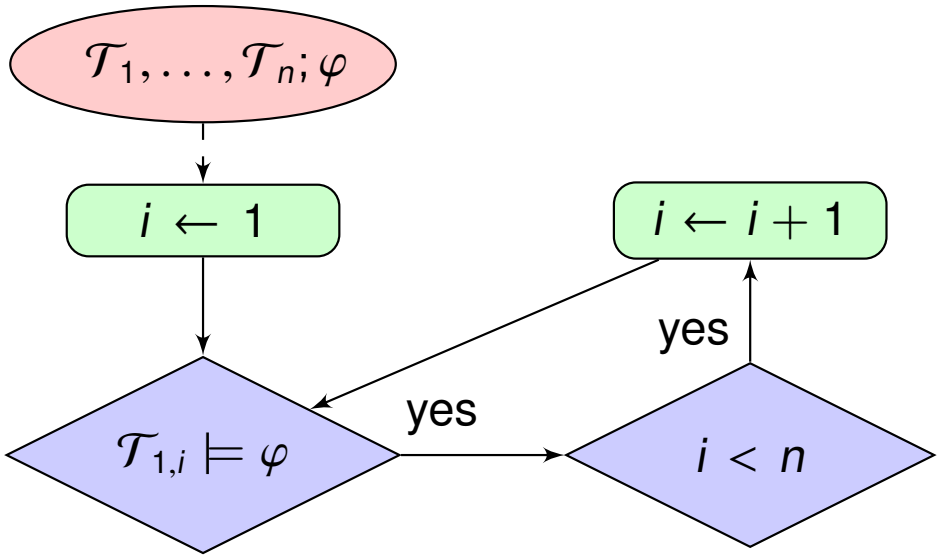

MA1
loop
λ =
{{Sync}}
A1
Sync
Sync
A4
A5
MA2
loop
λ =
{{Obs}, {Sync}}
Sync
obs
MA3
loop
λ =
{{sync}}
A4
MB1
loop
λ =
{{Sync}}
B1
Sync
MB2
—
λ = {∅}
B3


MA1
loop
λ =
{{Sync}}
A1
Sync
Sync
MA2
loop
λ =
{{Obs}, {Sync}}
A4
A5
Sync
obs
MA3
loop
λ =
{{sync}}
A4
MB1
loop
λ =
{{Sync}}
B1
Sync
MB2
—
λ = {∅}
B3
Act1∩Act2
=
M1
loop
λ =
{{Sync}}
A1 B1
Sync
M2
loop
λ =
{{Obs}, ∅}
A4 B3
A5 B3
obs
M3
loop
λ = {∅}
A4 B3

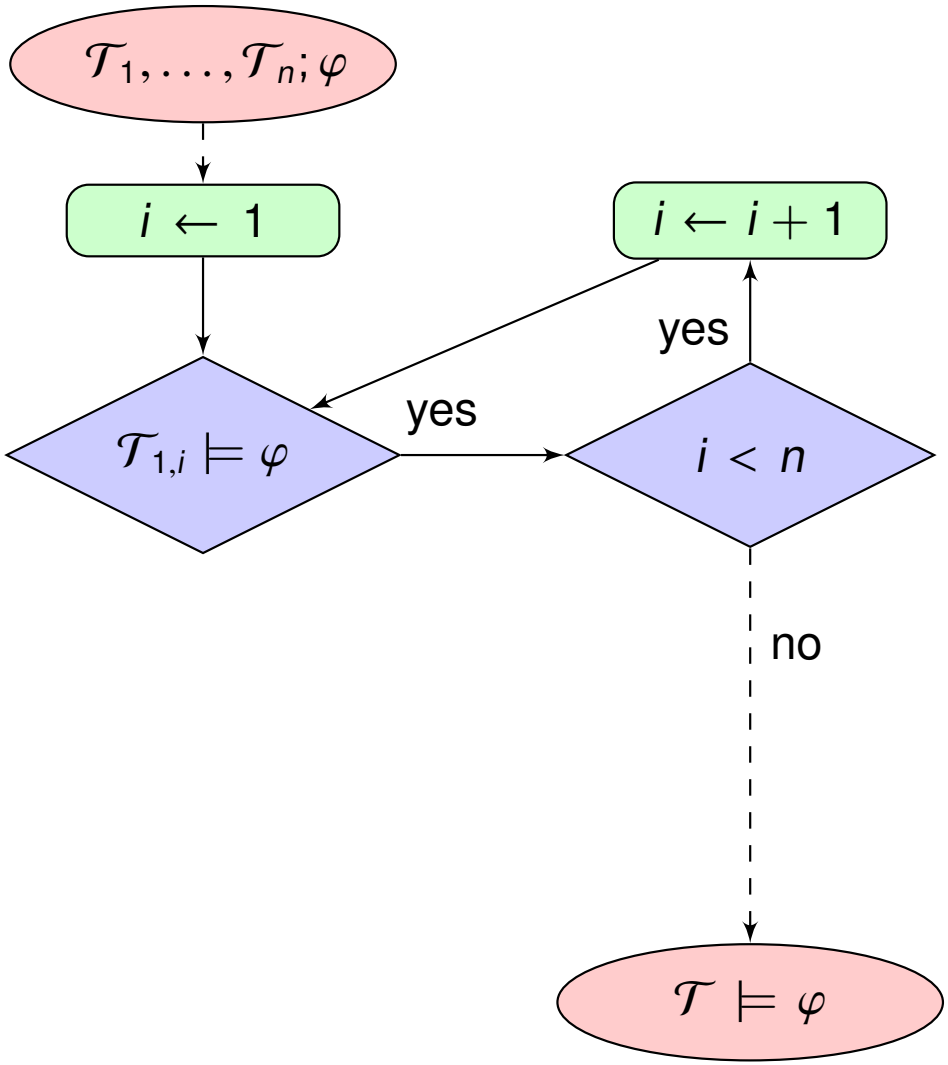
Incremental Verification Revisited



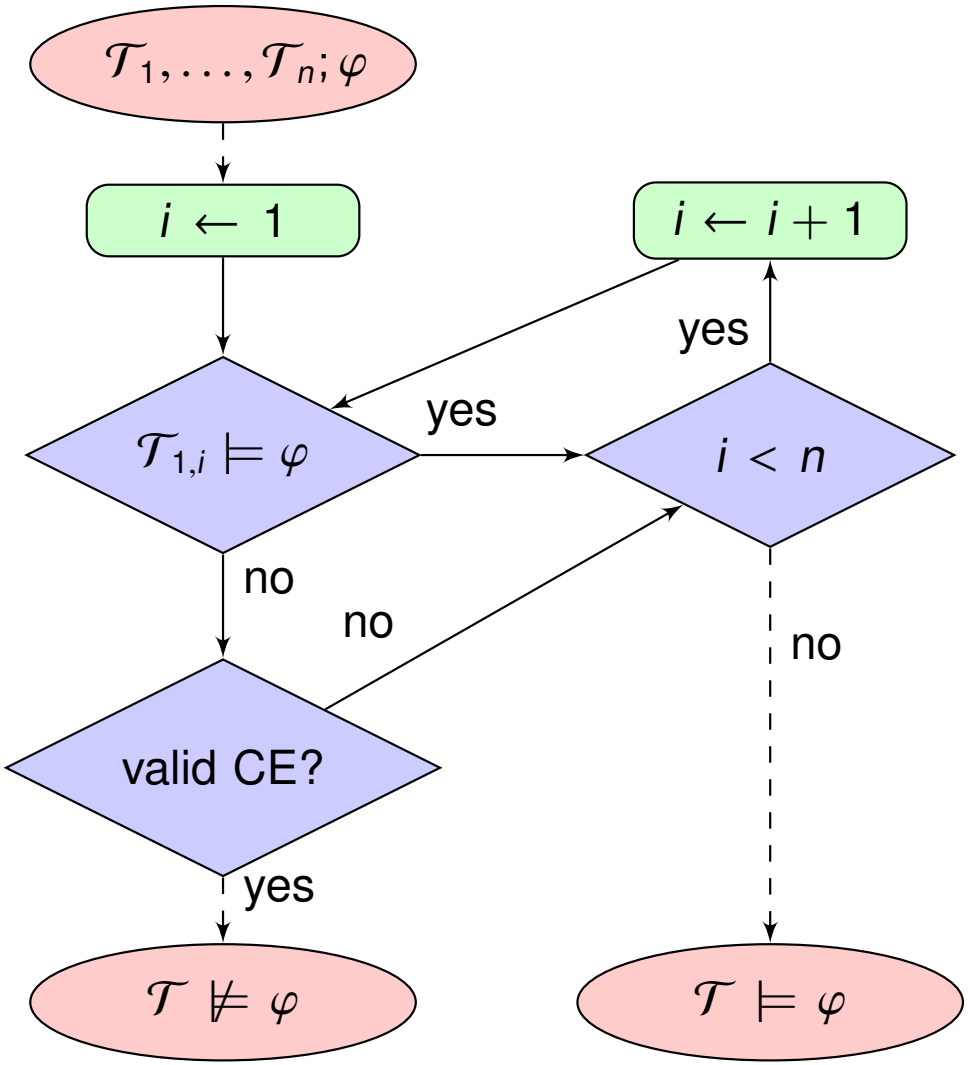
Incremental Verification Revisited



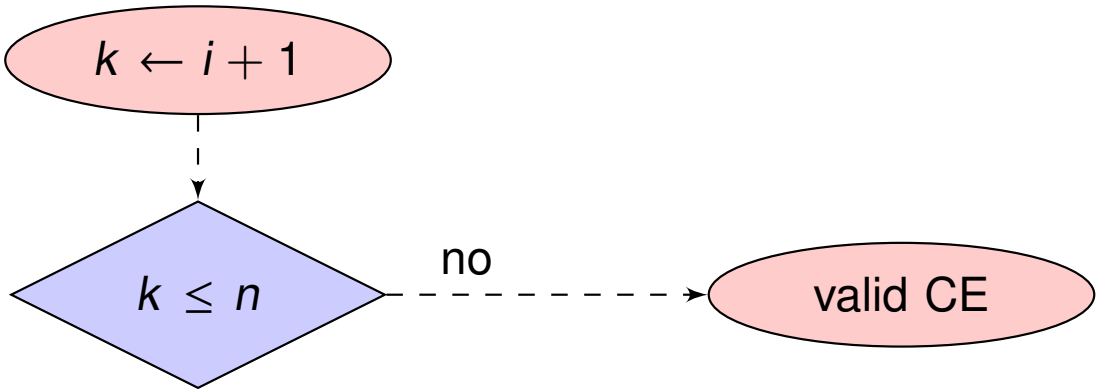
Incremental Verification Revisited



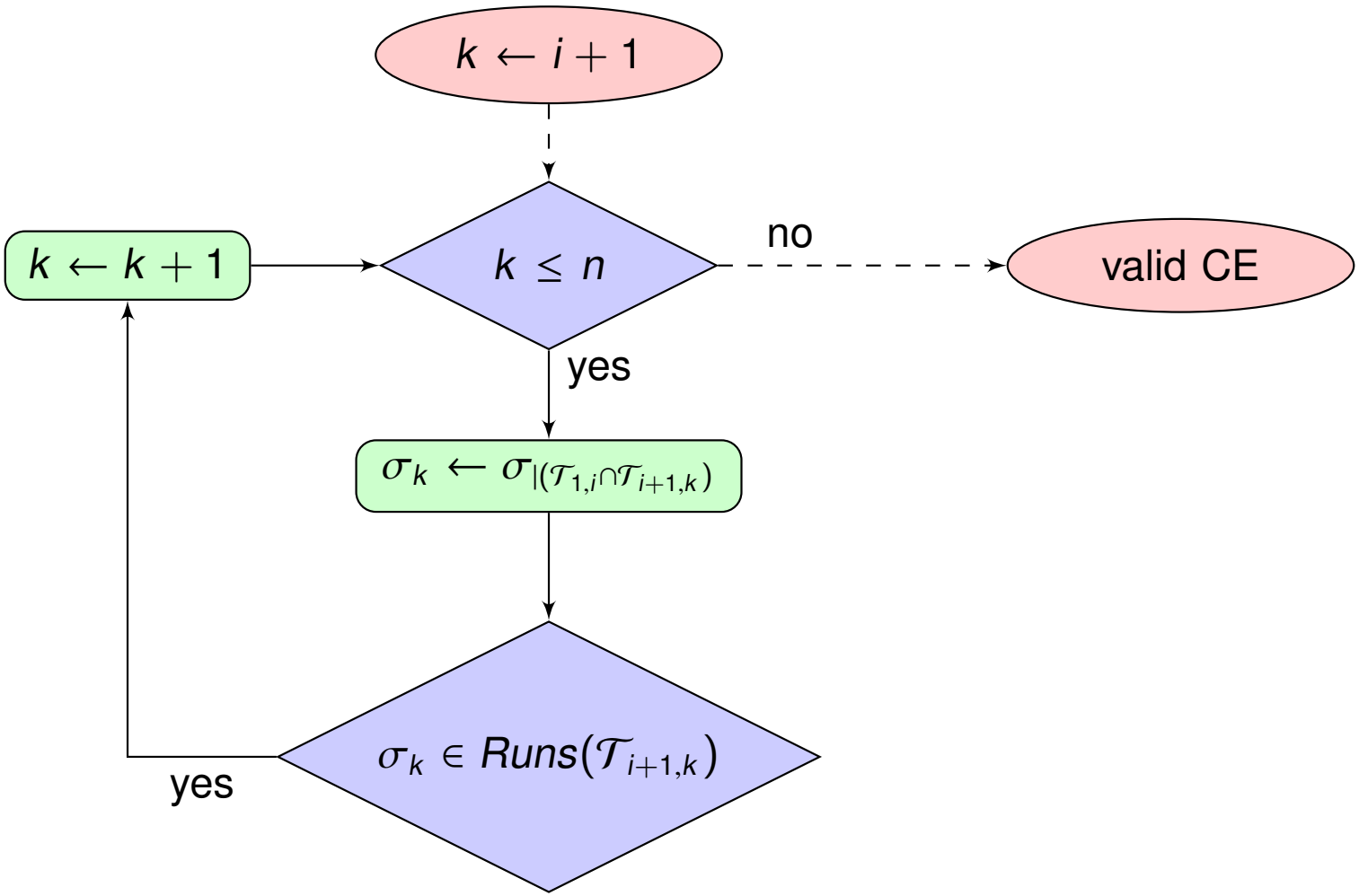
Incremental Verification Revisited



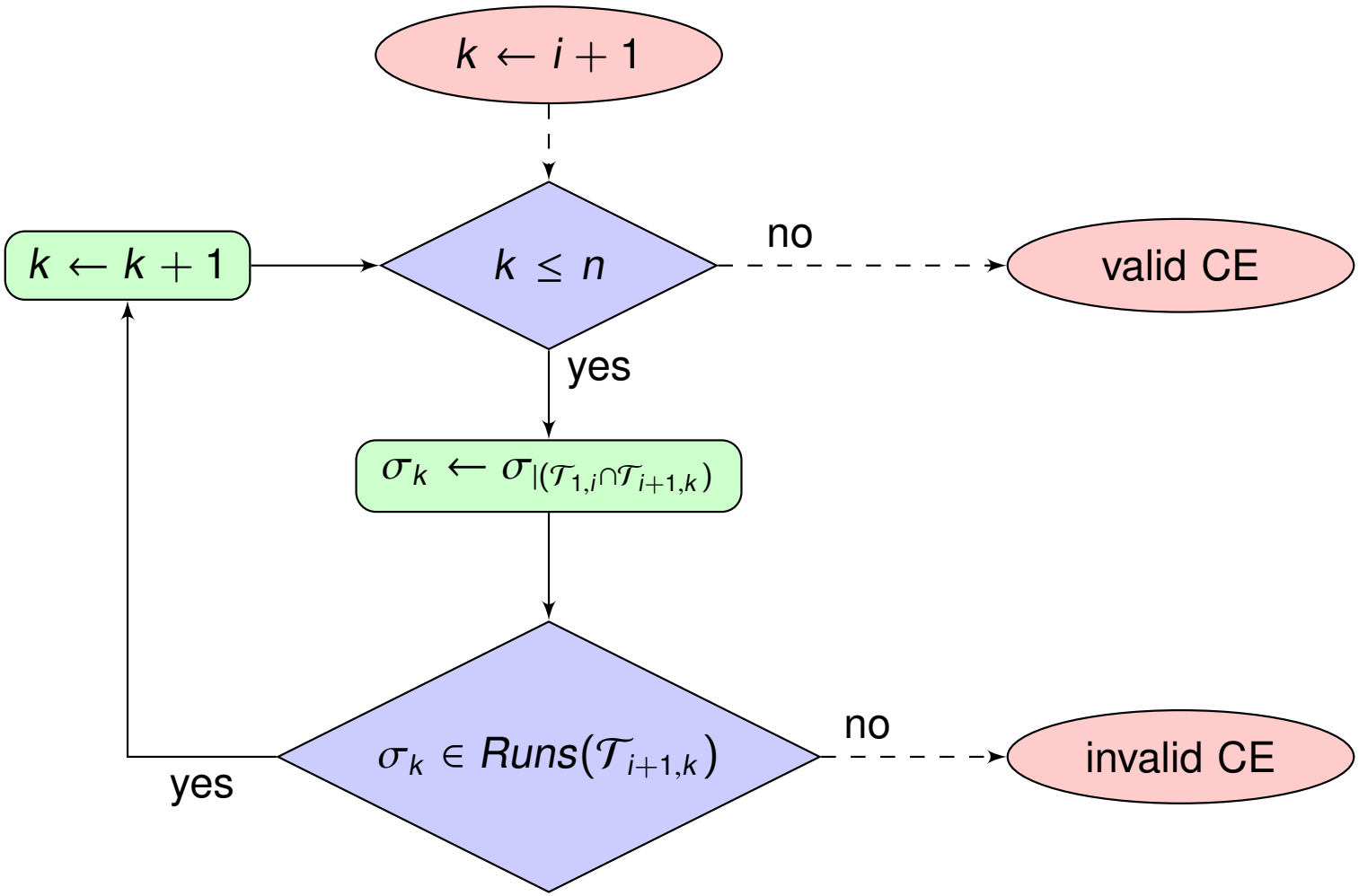
Checking Validity of Couterexamples



Checking Validity of Couterexamples



Checking Validity of Couterexamples



Summary and Advantages

Summary

- Incremental model-checking of LTL\X properties
- On-the-fly construction of a counterexample
- Compositional approach

Summary and Advantages

Summary

- Incremental model-checking of LTL\X properties
- On-the-fly construction of a counterexample
- Compositional approach

Advantages of the approach

- Reduce complexity with both incremental approach and use of SOGs
- Allow for black box components
- Reuse of components SOGs without additional computation

Perspectives

- Study the impact of components' structure on performances
- Software implementation of the different approaches
- Experiments on different kinds of case studies