Distributed Verification of Modular Systems

Mohand Cherif Boukala   Laure Petrucci

LSI, USTHB
Algiers, Algeria

LIPN, University Paris 13
Villetaneuse, France

Motivation

Exhaustive analysis of complex systems using state spaces
⇒ state space explosion problem

Coping with state space explosion

- Reduction of the state space
  - symmetries
  - partial orders
  - sleep sets, etc.
- Reduction of the representation
  - BDDs, DDDs, etc.
  - modular state spaces
  - distributed state spaces

Aims

Combine modular and distributed approaches

Motivation

Exhaustive analysis of complex systems using state spaces
⇒ state space explosion problem

Coping with state space explosion

- Reduction of the state space
  - symmetries
  - partial orders
  - sleep sets, etc.
- Reduction of the representation
  - BDDs, DDDs, etc.
  - modular state spaces
  - distributed state spaces

Aims

Combine modular and distributed approaches

Motivation

Exhaustive analysis of complex systems using state spaces
⇒ state space explosion problem

Coping with state space explosion

- Reduction of the state space
  - symmetries
  - partial orders
  - sleep sets, etc.
- Reduction of the representation
  - BDDs, DDDs, etc.
  - modular state spaces
  - distributed state spaces

Aims

Combine modular and distributed approaches

Motivation

Exhaustive analysis of complex systems using state spaces
⇒ state space explosion problem

Coping with state space explosion

- Reduction of the state space
  - symmetries
  - partial orders
  - sleep sets, etc.
- Reduction of the representation
  - BDDs, DDDs, etc.
  - modular state spaces
  - distributed state spaces

Aims

Combine modular and distributed approaches

Motivation

Exhaustive analysis of complex systems using state spaces
⇒ state space explosion problem

Coping with state space explosion

- Reduction of the state space
  - symmetries
  - partial orders
  - sleep sets, etc.
- Reduction of the representation
  - BDDs, DDDs, etc.
  - modular state spaces
  - distributed state spaces

Aims

Combine modular and distributed approaches

Outline

1. Characteristics of Modular Distributed state space generation
2. Properties verification
3. Experiments
4. Conclusion and future work

State space generation

Modular state spaces

Local state spaces

- specific to a module
- only local behaviour

Synchronisation graph

- global behaviour (fused transitions)
- nodes represent sets of states linked by local actions

Distributed architecture

Coordinator
- initiates the computation
- handles termination

Workers
- compute part of the state space
- collaborate via message passing

State space generation

Modular state spaces

Local state spaces

- specific to a module
- only local behaviour

Synchronisation graph

- global behaviour (fused transitions)
- nodes represent sets of states linked by local actions

Distributed architecture

Coordinator	initiates the computation handles termination
Workers	compute part of the state space collaborate via message passing

Modular distributed state space generation

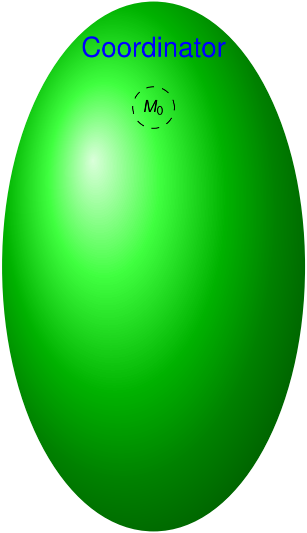
Coordinator

- builds the synchronisation graph
- coordinates the worker processes
- ensures termination

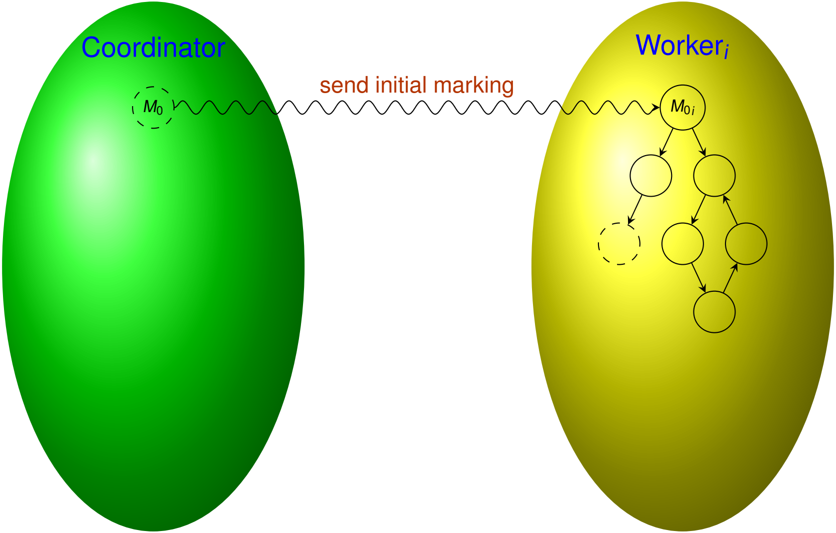
Workers

- constructs the local state space
- sends possible synchronisation points to the coordinator

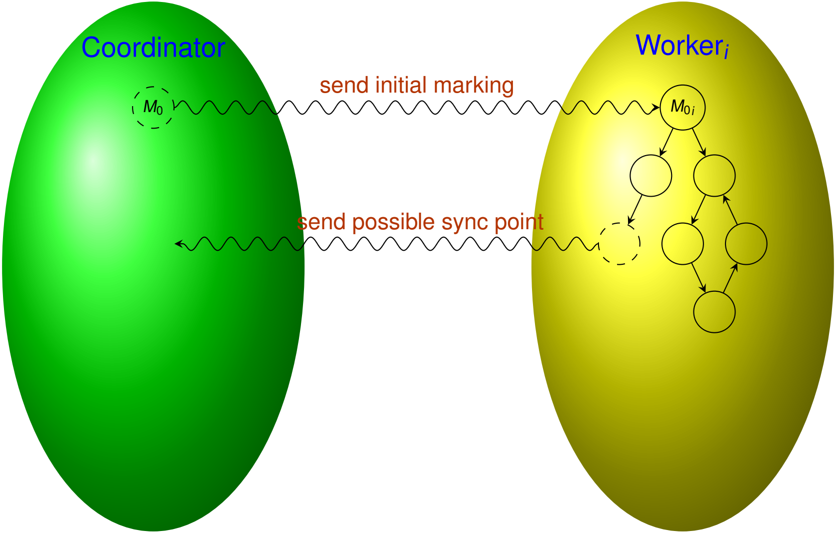
Distributed state space generation



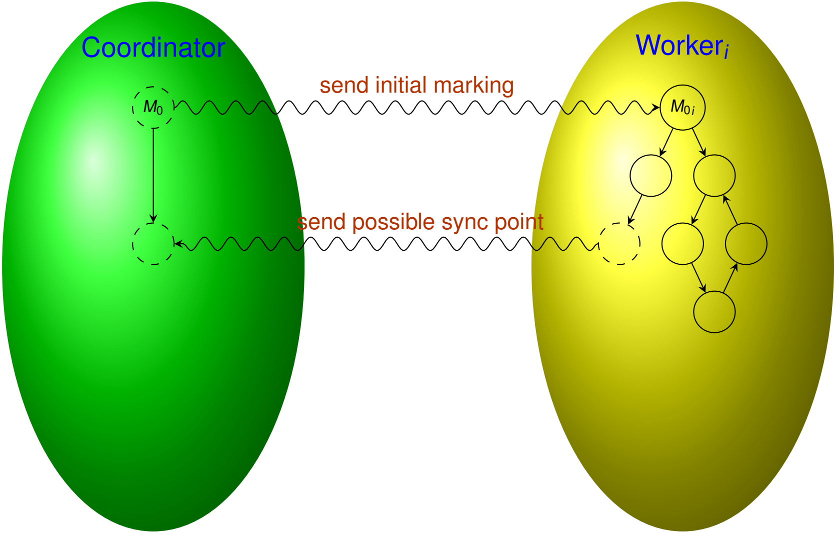
Distributed state space generation



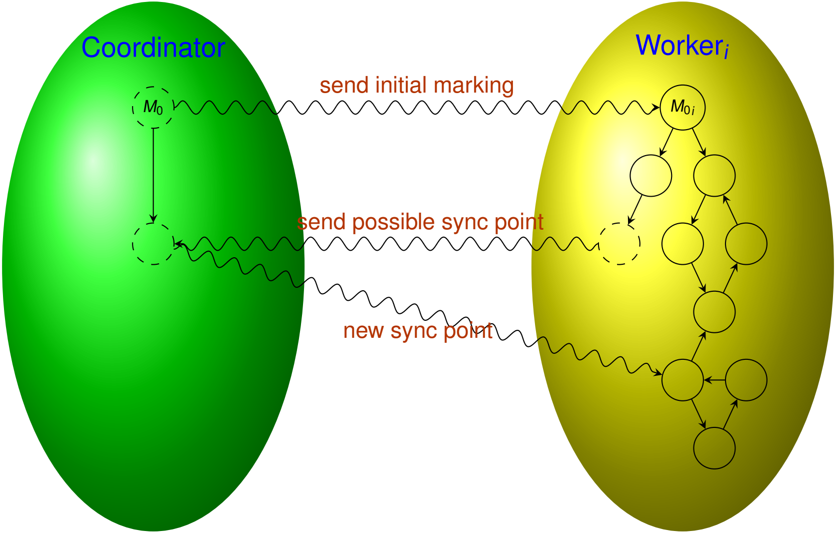
Distributed state space generation



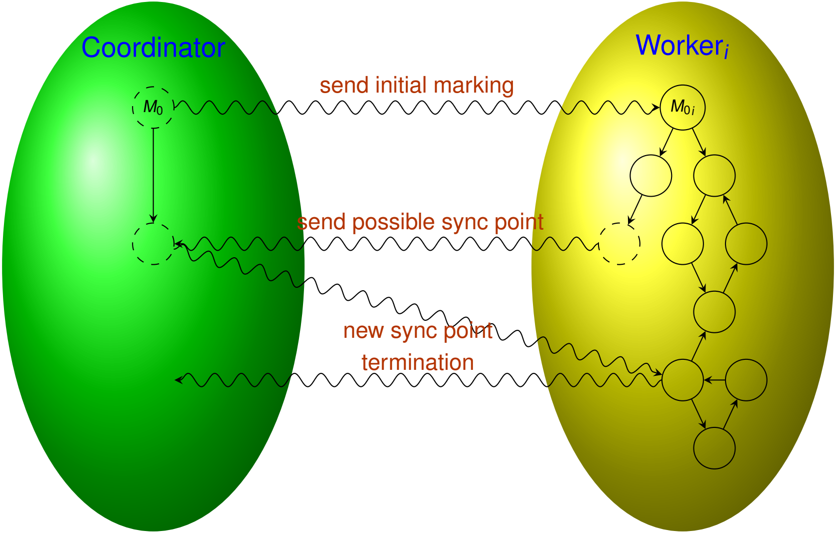
Distributed state space generation



Distributed state space generation



Distributed state space generation



Main characteristics

SCCs of local state spaces are updated during the construction

Messages contains the fused transition enabling and the SCCs of its markings

Synchronisation only focuses on participating modules

Termination occurs when all workers have finished computing their local state space, and there is no new synchronisation point

Verifying properties

- as much local computation as possible
- minimise the number of messages exchanged by worker processes

Reachability

Global part (Coordinator)

- sends partial markings to the worker processes
- If it receives a negative answer the marking is not reachable
- otherwise find a combination of the ancestor SCCs in the synchronisation graph

Local part (Worker processes)

- search for their partial marking in their local state space
- If it is not found the marking is not reachable
- otherwise send its ancestor SCCs to the coordinator

Reachability

Global part (Coordinator)

- sends partial markings to the worker processes
- If it receives a negative answer the marking is not reachable
- otherwise find a combination of the ancestor SCCs in the synchronisation graph

Local part (Worker processes)

- search for their partial marking in their local state space
- If it is not found the marking is not reachable
- otherwise send its ancestor SCCs to the coordinator

Reachability

Global part (Coordinator)

- sends partial markings to the worker processes
- If it receives a negative answer the marking is not reachable
- otherwise find a combination of the ancestor SCCs in the synchronisation graph

Local part (Worker processes)

- search for their partial marking in their local state space
- If it is not found the marking is not reachable
- otherwise send its ancestor SCCs to the coordinator

Deadlocks

Global part (Coordinator)

- find a combination of dead markings received on the arcs of the synchronisation graph
- If it does not label an arc but is reachable, then it is a deadlock

Local part (Worker processes)

- search for dead markings in their local state space
- If there is none the system is deadlock-free
- otherwise send them to the coordinator

Deadlocks

Global part (Coordinator)

- find a combination of dead markings received on the arcs of the synchronisation graph
- If it does not label an arc but is reachable, then it is a deadlock

Local part (Worker processes)

- search for dead markings in their local state space
- If there is none the system is deadlock-free
- otherwise send them to the coordinator

Liveness — Fused transition *tf*

Global part (Coordinator)

- If there exists a terminal SCC in the synchronisation graph not containing *tf* then *tf* is not live
- otherwise send nodes of the synchronisation graph to the worker processes
- if a combination of nodes received does not label an arc in the synchronisation graph, *tf* is not live

Local part (Worker processes)

- receive v_s
- send terminal SCCs reachable from v_s to the coordinator

Liveness — Fused transition *tf*

Global part (Coordinator)

- If there exists a terminal SCC in the synchronisation graph not containing *tf* then *tf* is not live
- otherwise send nodes of the synchronisation graph to the worker processes
- if a combination of nodes received does not label an arc in the synchronisation graph, *tf* is not live

Local part (Worker processes)

- receive v_s
- send terminal SCCs reachable from v_s to the coordinator

Liveness — Fused transition *tf*

Global part (Coordinator)

- If there exists a terminal SCC in the synchronisation graph not containing *tf* then *tf* is not live
- otherwise send nodes of the synchronisation graph to the worker processes
- if a combination of nodes received does not label an arc in the synchronisation graph, *tf* is not live

Local part (Worker processes)

- receive v_s
- send terminal SCCs reachable from v_s to the coordinator

Liveness — Local transition t

Global part (Coordinator)

- send nodes of the synchronisation graph to the worker processes
- if a combination of nodes received does not label an arc in the synchronisation graph, t is not live

Local part (Worker processes)

- receive v_s
- identify terminal SCCs that do not enable t
- send those reachable from v_s to the coordinator

Liveness — Local transition t

Global part (Coordinator)

- send nodes of the synchronisation graph to the worker processes
- if a combination of nodes received does not label an arc in the synchronisation graph, t is not live

Local part (Worker processes)

- receive v_s
- identify terminal SCCs that do not enable t
- send those reachable from v_s to the coordinator

Liveness — Local transition *t*

Global part (Coordinator)

- send nodes of the synchronisation graph to the worker processes
- if a combination of nodes received does not label an arc in the synchronisation graph, t is not live

Local part (Worker processes)

- receive v_s
- identify terminal SCCs that do not enable t
- send those reachable from v_s to the coordinator

Experimental results

Setting for experiments

- 1 machine for the coordinator process
- 11 for the worker processes
- philosophers and AGVs examples

Analysis of results

- significant gain in time during the construction
- few messages exchanged for the construction and reachability properties
- optimisation for liveness and home states, so as to decrease the number of messages

Experimental results

Setting for experiments

- 1 machine for the coordinator process
- 11 for the worker processes
- philosophers and AGVs examples

Analysis of results

- significant gain in time during the construction
- few messages exchanged for the construction and reachability properties
- optimisation for liveness and home states, so as to decrease the number of messages

Conclusion & Future work

Summary

- distributed modular state spaces
- distributed modular analysis

Future work

- apply to larger case studies
- extension to temporal logic properties

Conclusion & Future work

Summary

- distributed modular state spaces
- distributed modular analysis

Future work

- apply to larger case studies
- extension to temporal logic properties