Towards parallel verification of concurrent systems using the Symbolic Observation Graph

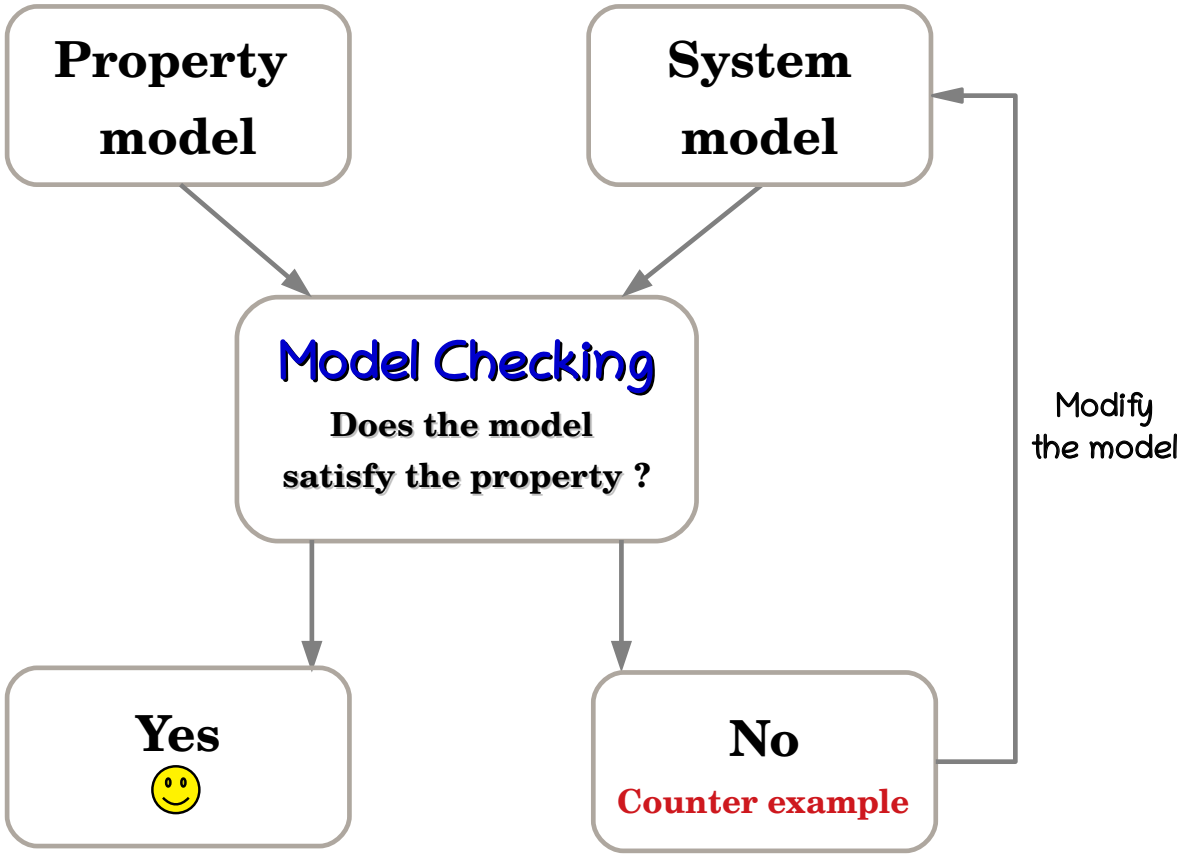
Hiba Ouni, Kais Klai, Chiheb Ameur Abid, Belhassen Zouari

LIPN Laboratory, University of Paris 13
University of Tunis el Manar

Outline

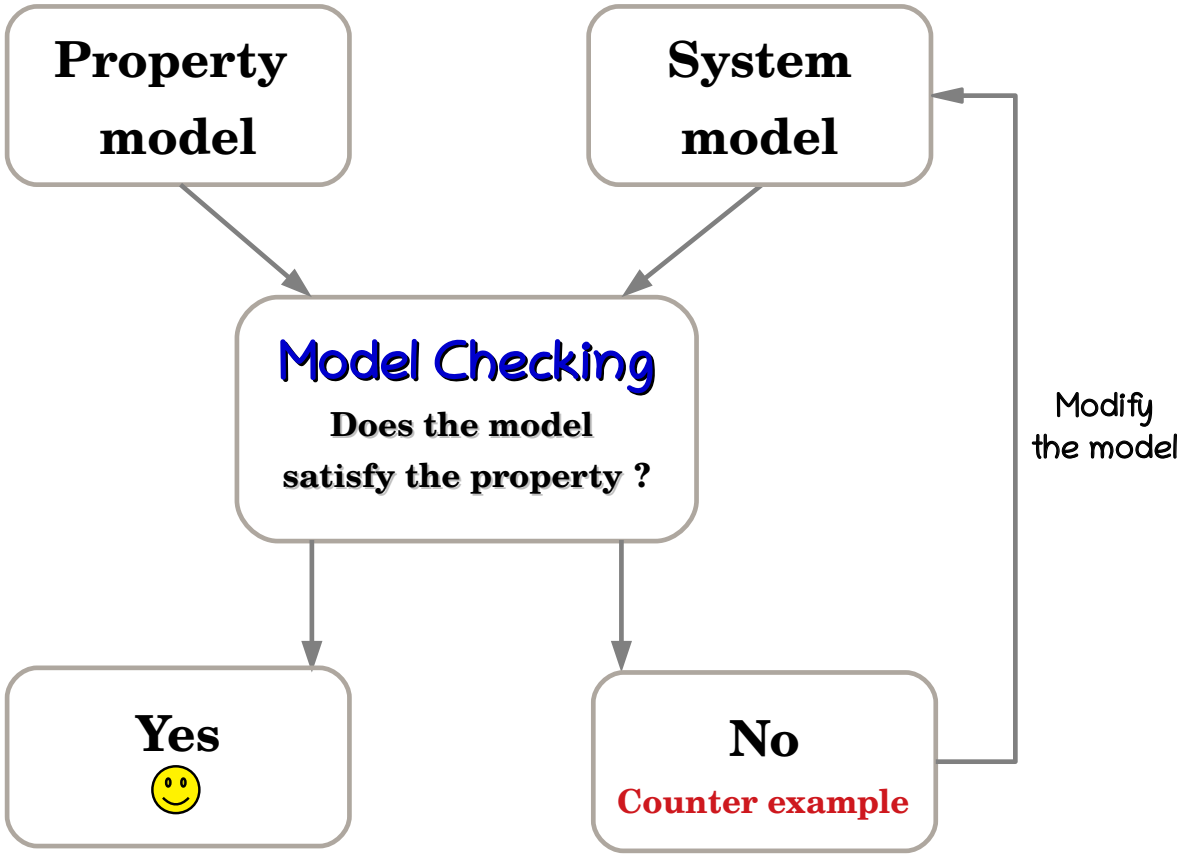
Outline

Context and motivation



- Model checking is based on state space traversal algorithms.
  ⇒ State explosion problem

Context and motivation



- Model checking is based on state space traversal algorithms.
  $\Rightarrow$ State explosion problem

Motivation

Tackling the explosion problem

- **On-the-fly verification**
  - Generation of only the 'interesting' part of the state space.
- **Symbolic representation**
  - Compact representation of the system by using Decision Diagram techniques (eg. BDD/MDD)
- **Symbolic Observation Graph (SOG)**
  - Checks linear temporal logic (LTL) properties over finite systems.
- **Parallel and distributed approach**
  - Offers more available memory for storage of the state space.

Motivation

Tackling the explosion problem

- **On-the-fly verification**
  - Generation of only the 'interesting' part of the state space.
- **Symbolic representation**
  - Compact representation of the system by using Decision Diagram techniques (eg. BDD/MDD)
- **Symbolic Observation Graph (SOG)**
  - Checks linear temporal logic (LTL) properties over finite systems.
- **Parallel and distributed approach**
  - Offers more available memory for storage of the state space.

Motivation

Tackling the explosion problem

- **On-the-fly verification**
  - Generation of only the 'interesting' part of the state space.
- **Symbolic representation**
  - Compact representation of the system by using Decision Diagram techniques (eg. BDD/MDD)
- **Symbolic Observation Graph (SOG)**
  - Checks linear temporal logic (LTL) properties over finite systems.
- **Parallel and distributed approach**
  - Offers more available memory for storage of the state space.

Motivation

Tackling the explosion problem

- **On-the-fly verification**
  - Generation of only the 'interesting' part of the state space.
- **Symbolic representation**
  - Compact representation of the system by using Decision Diagram techniques (eg. BDD/MDD)
- **Symbolic Observation Graph (SOG)**
  - Checks linear temporal logic (LTL) properties over finite systems.
- **Parallel and distributed approach**
  - Offers more available memory for storage of the state space.

Motivation

Tackling the explosion problem

- **On-the-fly verification**
  - Generation of only the 'interesting' part of the state space.
- **Symbolic representation**
  - Compact representation of the system by using Decision Diagram techniques (eg. BDD/MDD)
- **Symbolic Observation Graph (SOG)**
  - Checks linear temporal logic (LTL) properties over finite systems.
- **Parallel and distributed approach**
  - Offers more available memory for storage of the state space.

Outline

Symbolic Observation Graph (SOG)

Aim

- Check properties on an abstraction of the state space.

Principle

- Observe the actions occurring in an LTL formula.

$$\text{f : action-based formula} \Rightarrow \text{Observed transitions}$$
$$G(t_1 \Rightarrow F t_2) \rightarrow \quad T_{Obs} = \{t_1, t_2\}$$

- Ignore the unobserved behavior (except deadlock and livelock)
- Keep the state changes due to observation

Symbolic Observation Graph (SOG)

Aim

- Check properties on an abstraction of the state space.

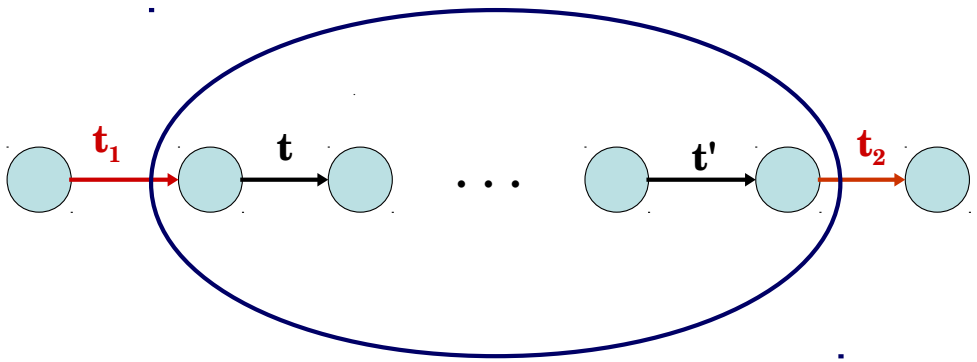
Principle

- Observe the actions occurring in an LTL formula.

f : action-based formula $\Rightarrow$ Observed transitions

$$G(t_1 \Rightarrow F t_2) \rightarrow \quad T_{Obs} = \{t_1, t_2\}$$

- Ignore the unobserved behavior (except deadlock and livelock)
- Keep the state changes due to observation



Symbolic Observation Graph (SOG)

Aim

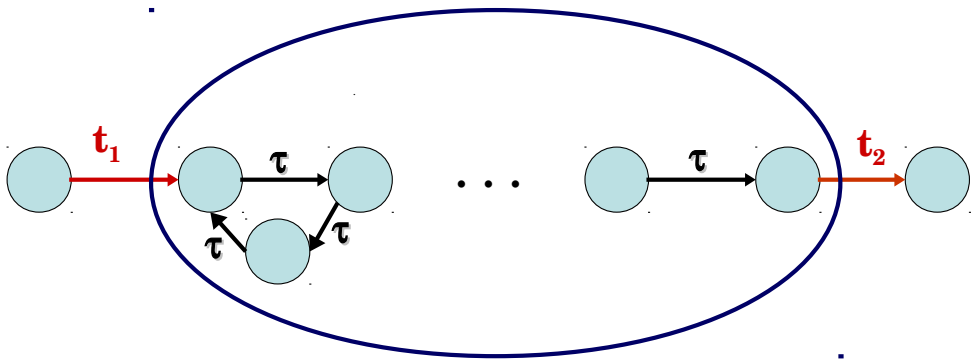
- Check properties on an abstraction of the state space.

Principle

- Observe the actions occurring in an LTL formula.

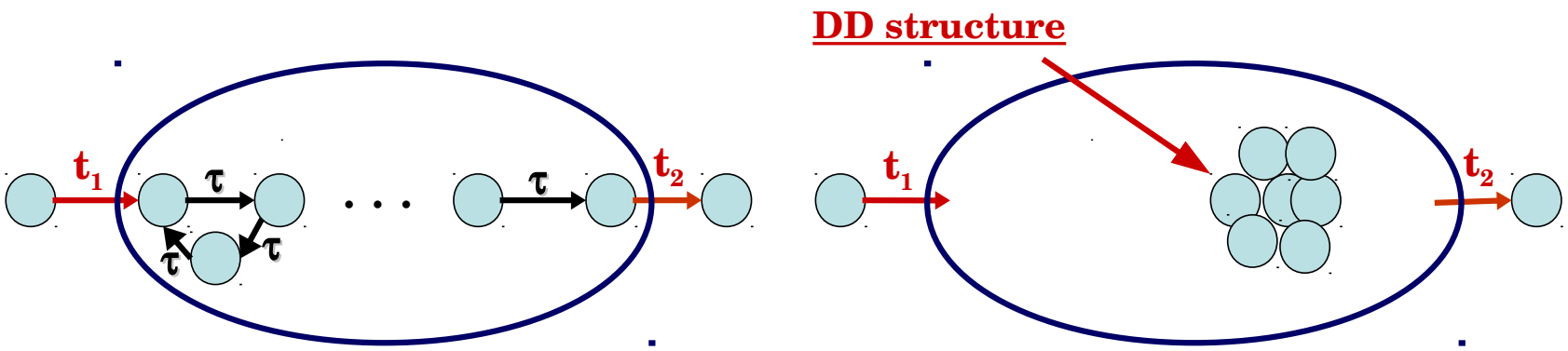
$$\text{f : action-based formula} \Rightarrow \text{Observed transitions}$$
$$G(t_1 \Rightarrow F t_2) \rightarrow \quad T_{Obs} = \{t_1, t_2\}$$

- Ignore the unobserved behavior (except deadlock and livelock)
- Keep the state changes due to observation



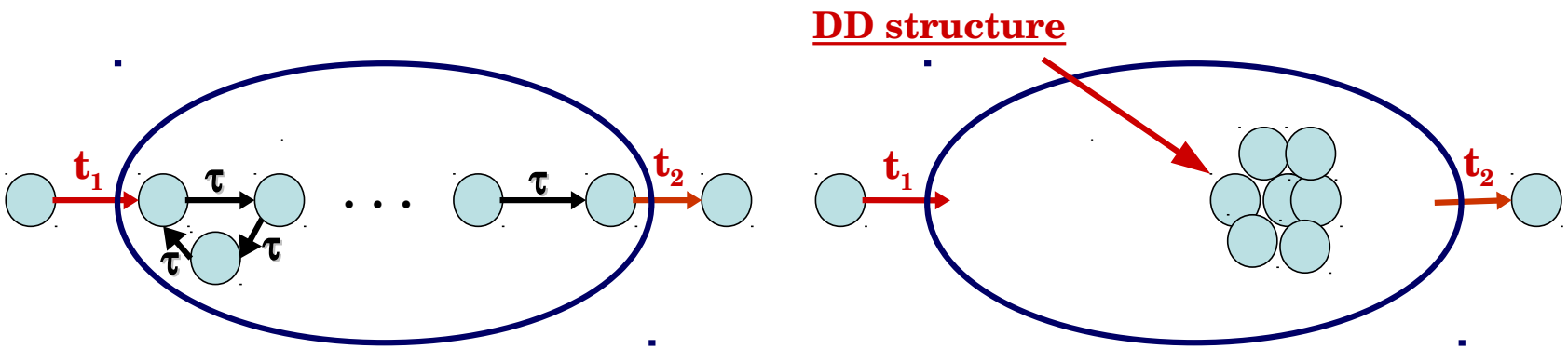
Symbolic Observation Graph (SOG)

- The nodes of the SOG (aggregates) are defined as sets of states linked with unobserved transitions
  - encoded symbolically with BDDs/MDDs.
- The edges are labeled with observed transitions only
  - represented explicitly.



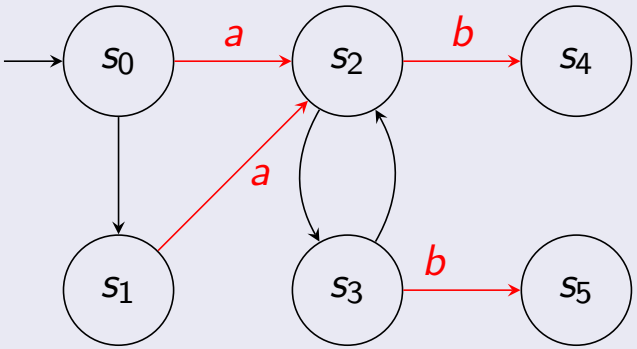
Symbolic Observation Graph (SOG)

- The nodes of the SOG (aggregates) are defined as sets of states linked with unobserved transitions
  - encoded symbolically with BDDs/MDDs.
- The edges are labeled with observed transitions only
  - represented explicitly.

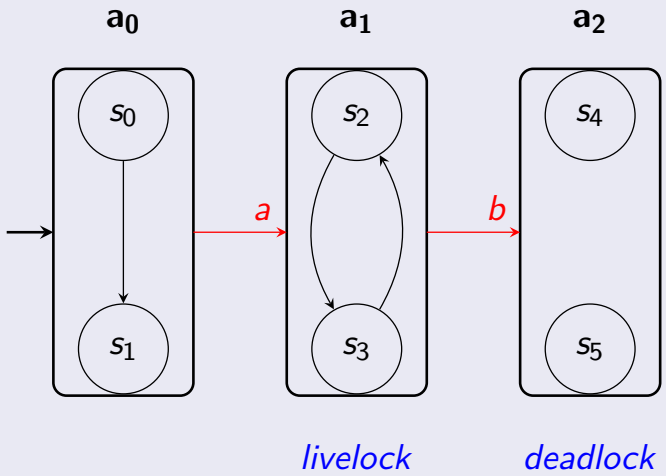


Symbolic Observation Graph (SOG)

Example of an LTS



A SOG with $Obs = \{a, b\}$



Symbolic Observation Graph (SOG)


Klai, Kais and Poitrenaud, Denis
MC-SOG: An LTL model checker based on symbolic observation graphs
Petri Nets 2009, 288–306.


Duret-Lutz, Alexandre and Klai, Kais and Poitrenaud, Denis and Thierry-Mieg, Yann
International Symposium on Automated Technology for Verification and Analysis
ATVA 2011: 336–350.

Outline

Hybrid approach

Multithreaded approach

- Shared memory
  √ Dynamic load balancing scheme
  × Limitation of the number of processors
  × Cost

Distributed approach

- Distributed memory
  √ Large number of processors
  × Static load balancing scheme
  × Communication overhead

Hybrid Approach

- Distributed shared memory
  √ Large number of multi-core processors (cluster)
  √ Static and dynamic load balancing

Hybrid approach

Multithreaded approach

- Shared memory
  - √ Dynamic load balancing scheme
  - × Limitation of the number of processors
  - × Cost

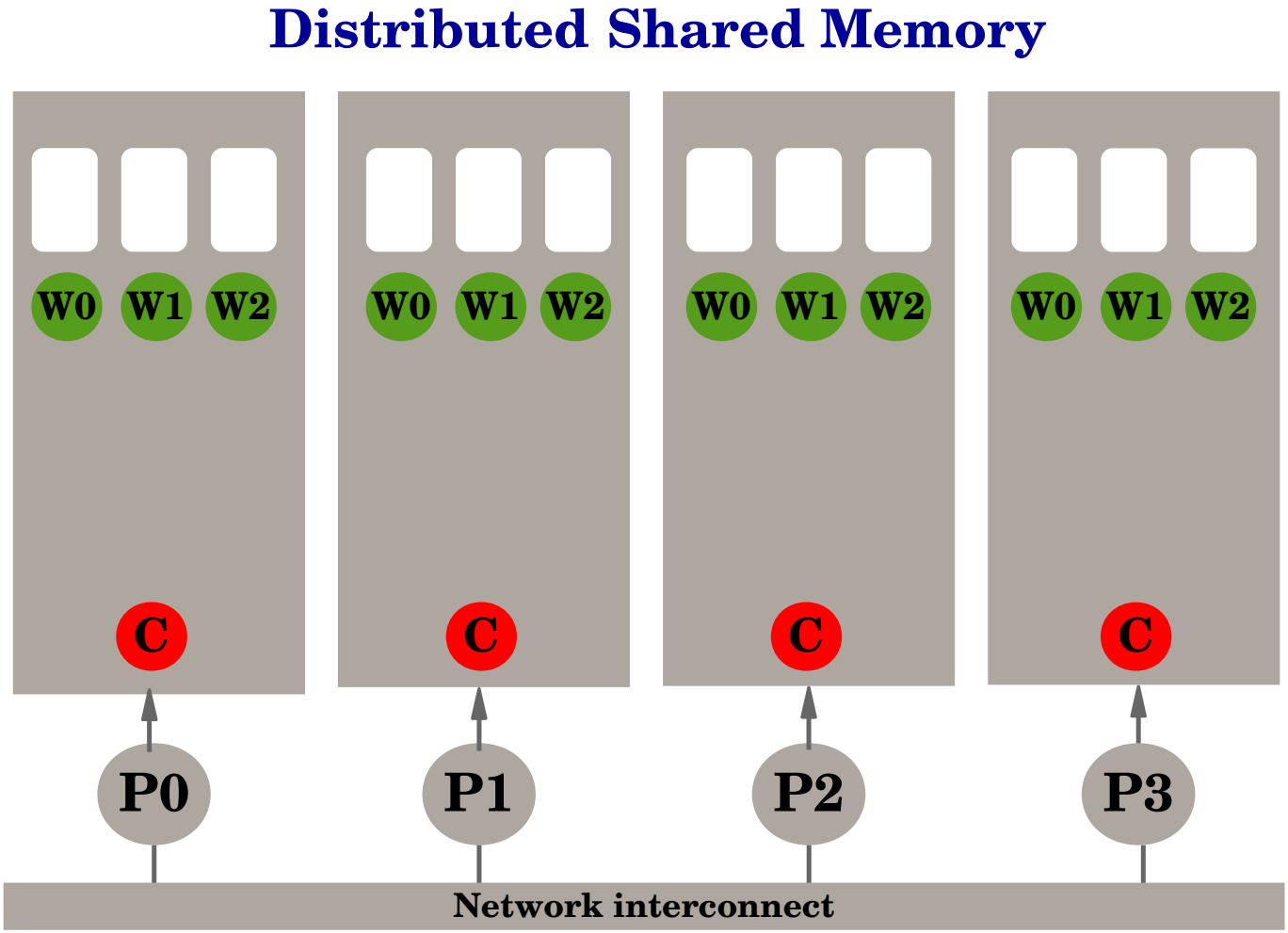
Distributed approach

- Distributed memory
  - √ Large number of processors
  - × Static load balancing scheme
  - × Communication overhead

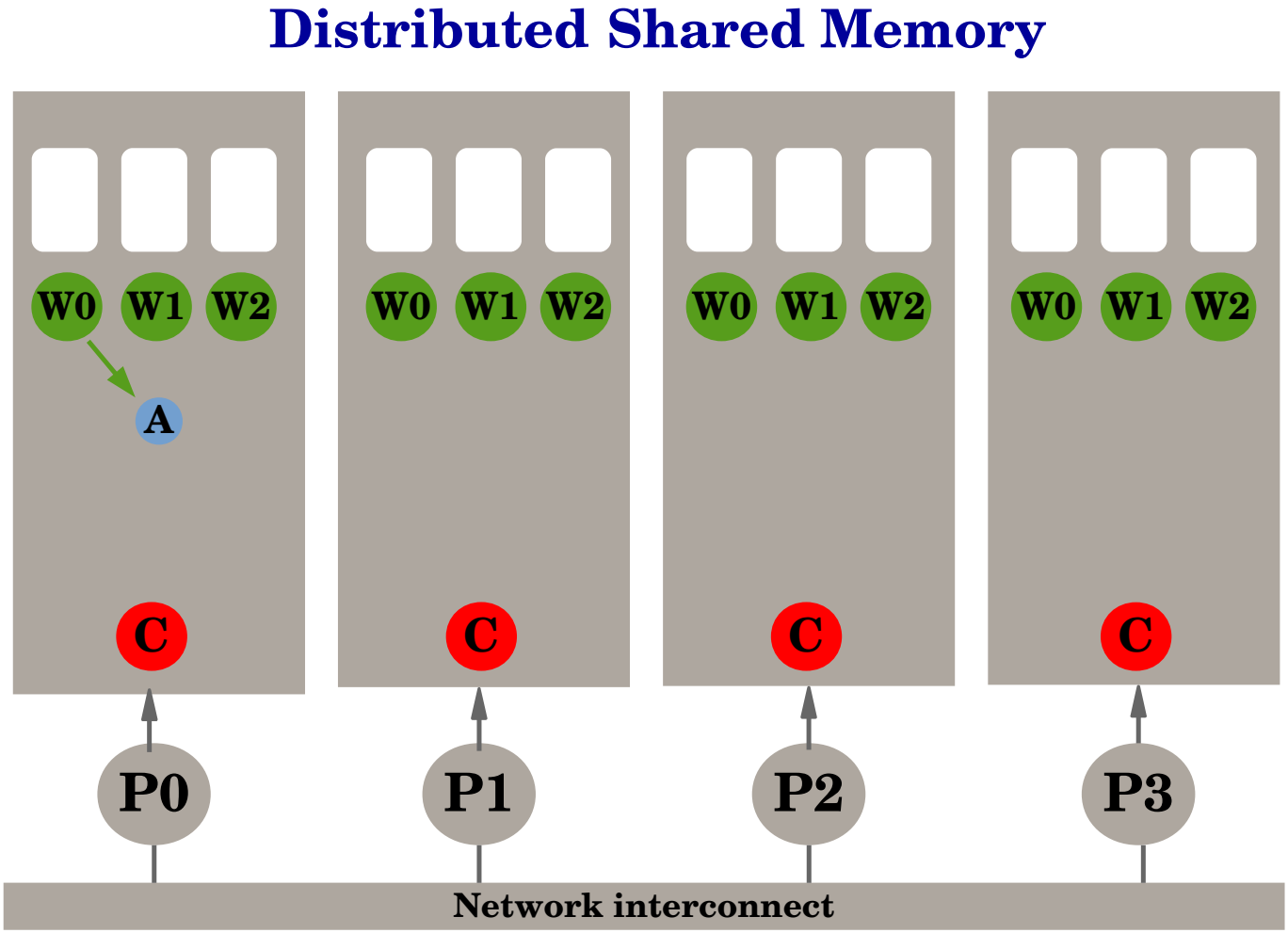
Hybrid Approach

- Distributed shared memory
  - √ Large number of multi-core processors (cluster)
  - √ Static and dynamic load balancing

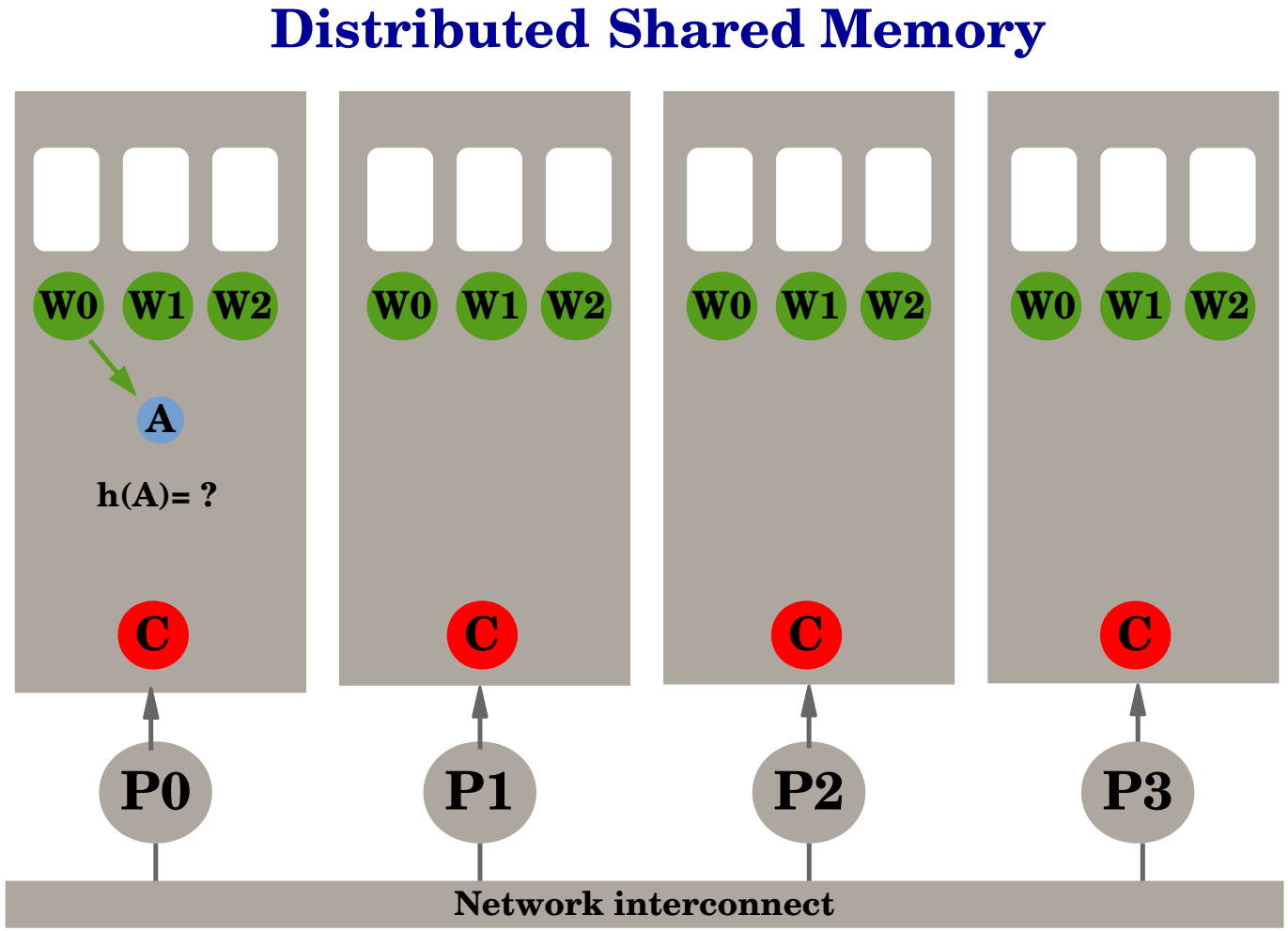
Hybrid approach



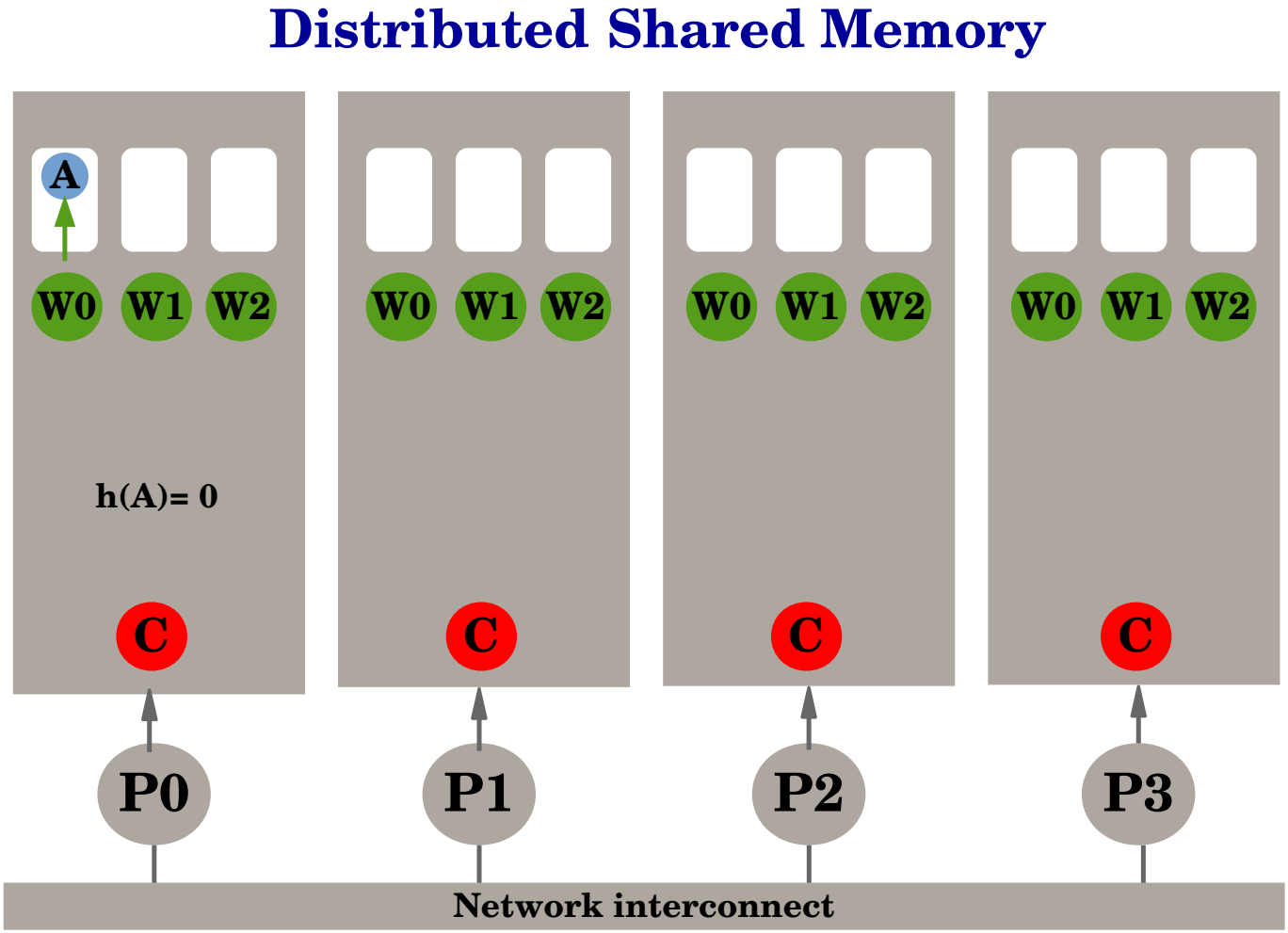
Hybrid approach



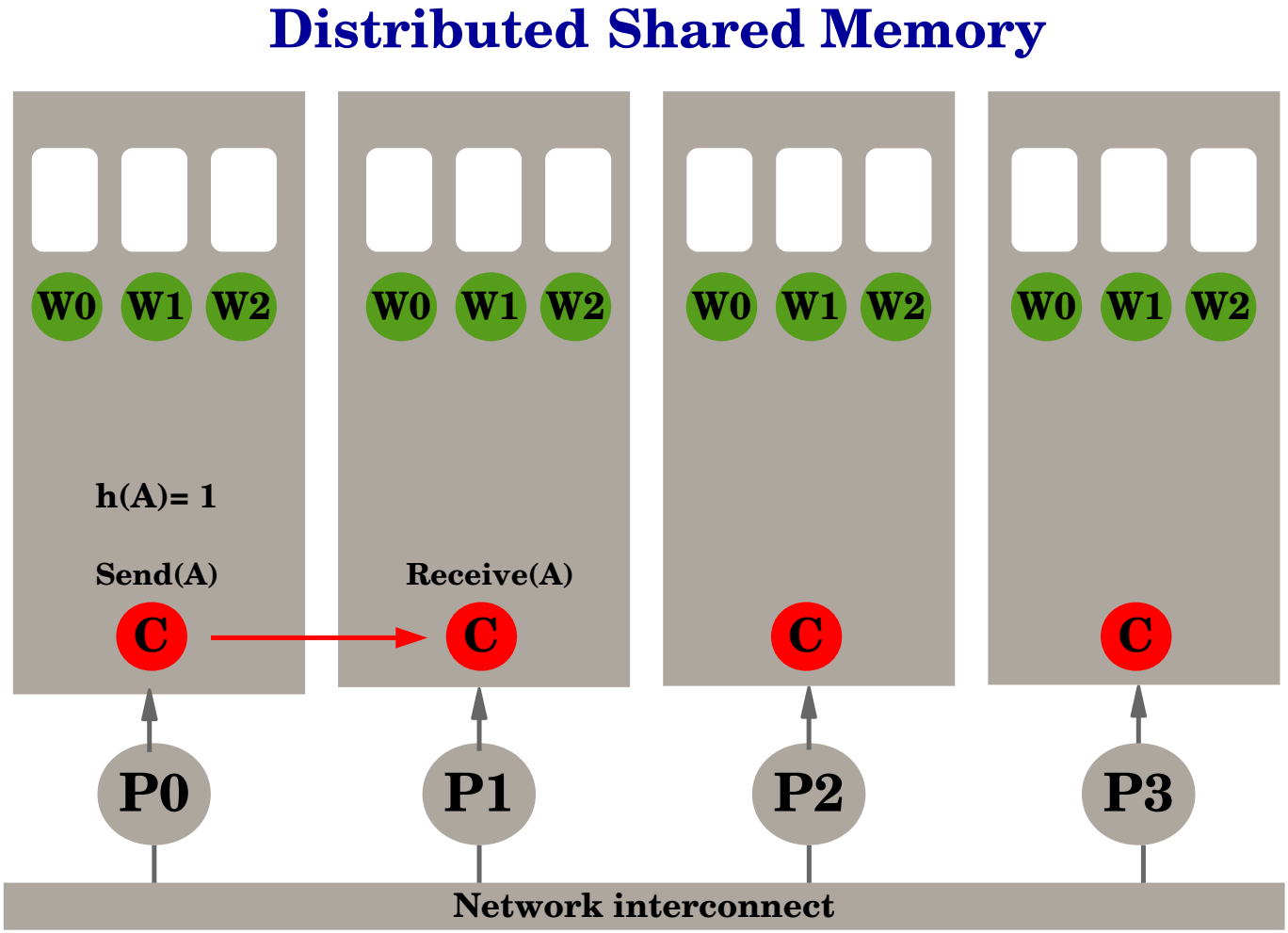
Hybrid approach



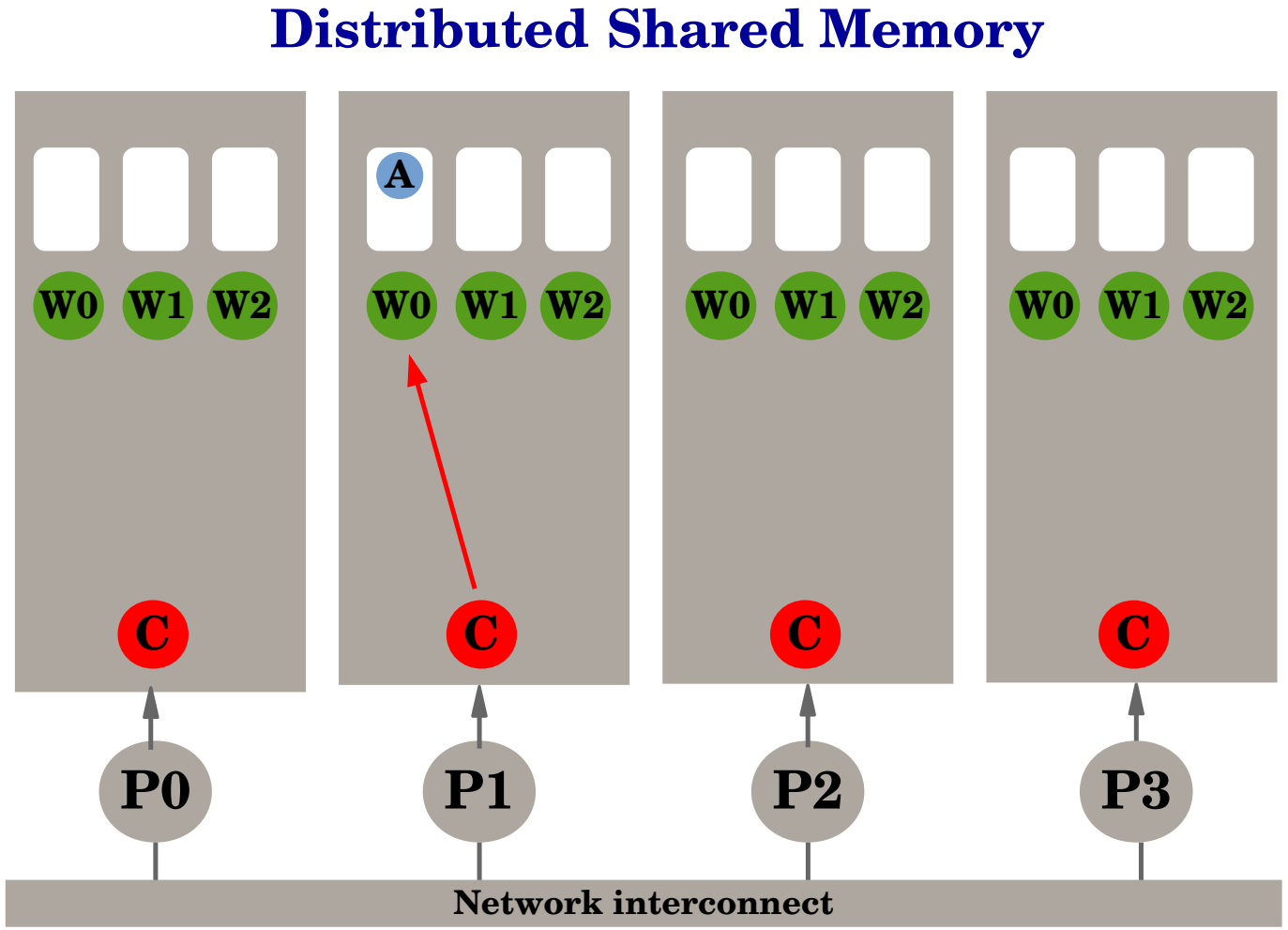
Hybrid approach



Hybrid approach



Hybrid approach



Outline

Hybrid VS Sequential

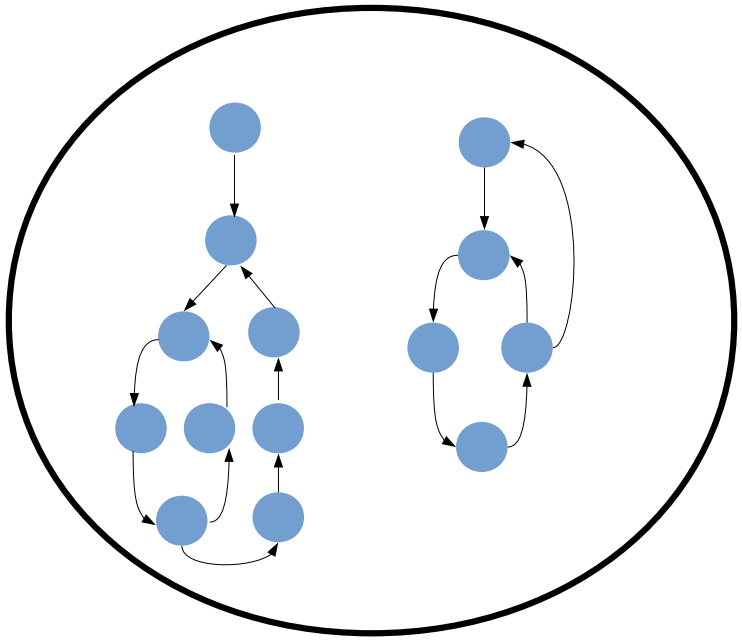
number of threads / process = 12

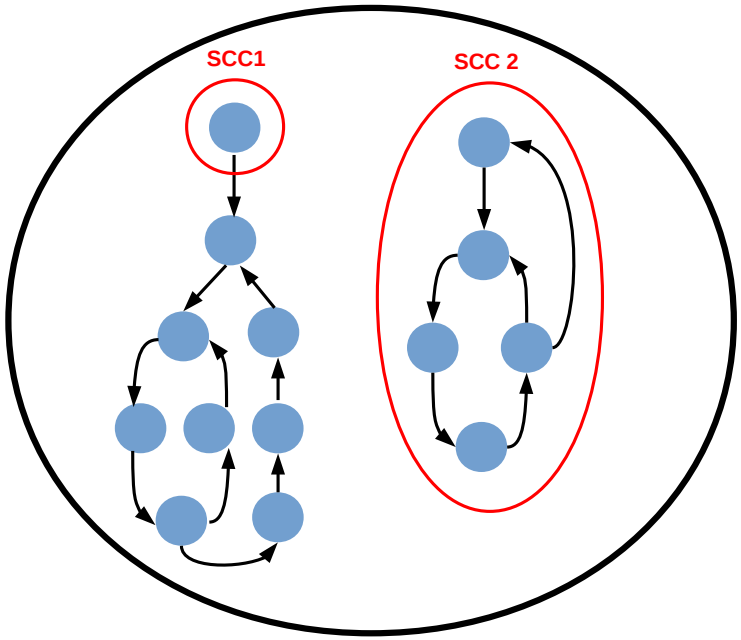
Net	Obs	states	Agg	Arcs	Seq	2	4	6	8	10	12	SpMax
ring4	8	5136	304	1280	2.15	0.57	0.68	0.70	0.70	0.77	0.71	3.7
ring5	10	53856	1632	8320	43.46	4.39	4.13	4.28	4.25	3.84	3.98	11.3
ring6	12	575296	3805	20698	870.87	71.34	37.16	28.51	23.44	21.05	18.66	46.6
philo8	16	103682	256	2048	11.94	1.07	0.94	1.11	1.14	1.21	1.29	12.7
philo10	20	1.86×10^6	1024	10240	311.10	22.62	19.23	17.18	14.42	13.91	14.58	22.3
fms4	4	438600	266	830	208.11	22.84	23.15	23.88	26.89	26.14	31.34	9.1
fms5	4	2.89×10^6	93280	519972	2088.75	502.95	473.68	406.51	343,22	481.73	524.64	6.0
robot4	6	48620	2574	11649	7.10	1.14	1.19	2.31	3.08	3.42	3.37	6.2
robot5	6	184756	6006	28600	36.16	4.06	3.28	4.63	4.77	5.98	5.73	11.0
robot6	6	587860	12376	61061	135.09	36.47	34.72	23.66	21.12	17.52	20.45	7.7
robot7	6	1.63×10^6	23256	117776	505.90	48.87	32.89	37.52	43.27	49.66	53.15	15.4
train2	4	86515	81	188	42.49	8.45	7.99	8.39	9.41	9.97	10.86	5.3
train2	6	86515	178	448	29.64	4.12	4.23	4.51	4.26	5.17	4.55	7.2
train2	8	86515	1660	6696	26.73	3.23	3.51	3.50	3.45	3.35	3.47	8.2
erk10	4	47047	505	1803	46.63	3.95	3.48	3.16	3.08	3.64	4.00	15.1
erk20	8	1.69×10^6	21230	15297	957.68	62.15	58.33	51.71	53.67	57.32	60.84	18.5

Distributed VS Hybrid VS Multithread

Model		Distributed		Hybrid		Multi-thread	
Net	Seq	runtime	SpMax	runtime	SpMax	runtime	SpMax
ring4	2.15	0.64	3.3	0.57	3.7	0.42	5.1
ring5	43.46	11.39	3.8	3.98	11.3	5.71	7.6
ring6	870.87	243.13	3.6	18.66	46.6	87.71	9.9
philo6	0.47	0.28	1.6	0.17	2.7	0.12	3.9
philo8	11.94	5.63	2.1	0.94	12.7	1.89	6.3
philo10	311.10	168.03	1.8	13.91	22.3	36.67	8.5
fms4	208.11	162.74	1.3	22.84	9.1	21.85	9.5
fms5	2088.75	1543.52	1.3	406.50	5.1	222.82	9.3
robot4	7.10	3.03	2.3	1.14	6.2	0.92	7.7
robot5	36.16	16.94	2.1	3.28	11.0	4.86	7.4
robot6	135.09	70.67	1.9	17.52	7.7	17.75	7.6
robot7	505.90	311.78	1.6	32.89	15.4	61.97	8.1
train2	42.49	28.88	1.4	7.99	5.3	12.58	3.3
train2	29.64	16.10	1.8	4.12	7.2	5.83	5.0
train2	26.73	10.73	2.5	3.23	8.2	2.86	9.0
erk10	46.63	18.84	2.4	3.08	15.1	5.11	9.1
erk20	957.68	397.23	2.4	51.71	18.5	86.44	11.0

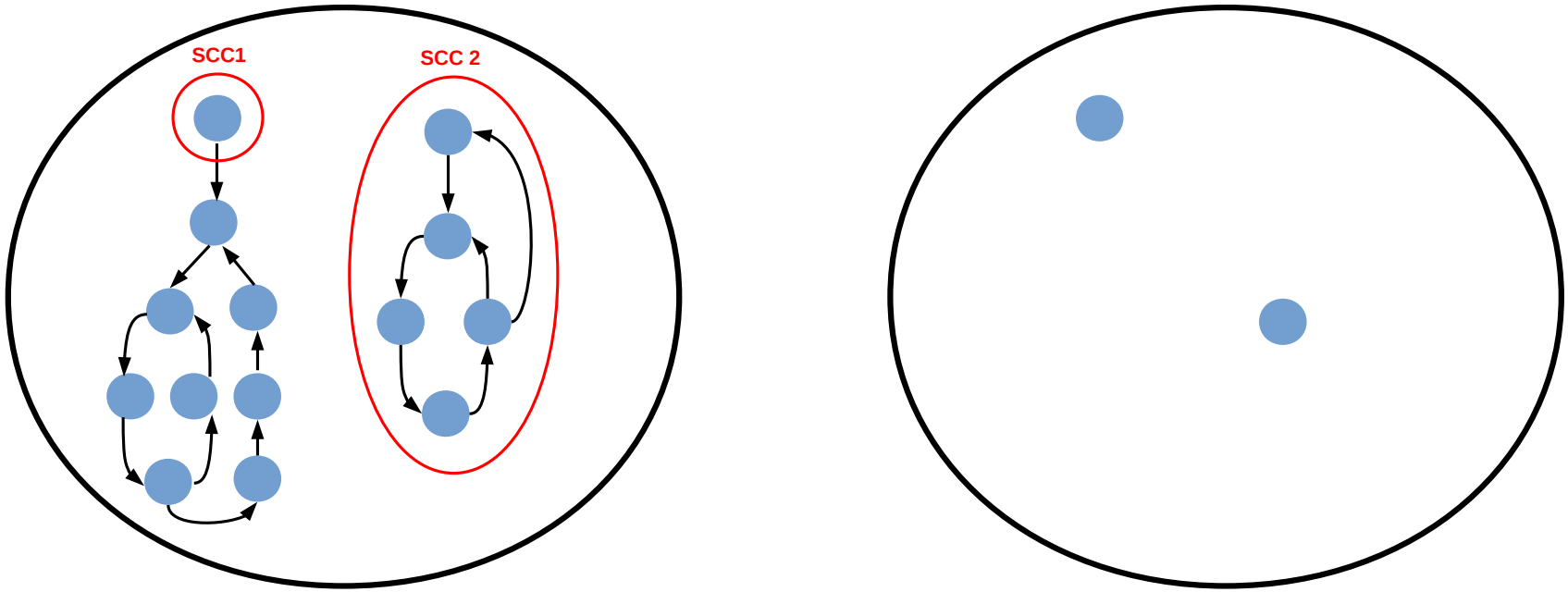
Canonicalization algorithm





- Reduce the set of states associated with each aggrgate.
- Determining a uniquely representative per each SCC source.

Canonicalization algorithm

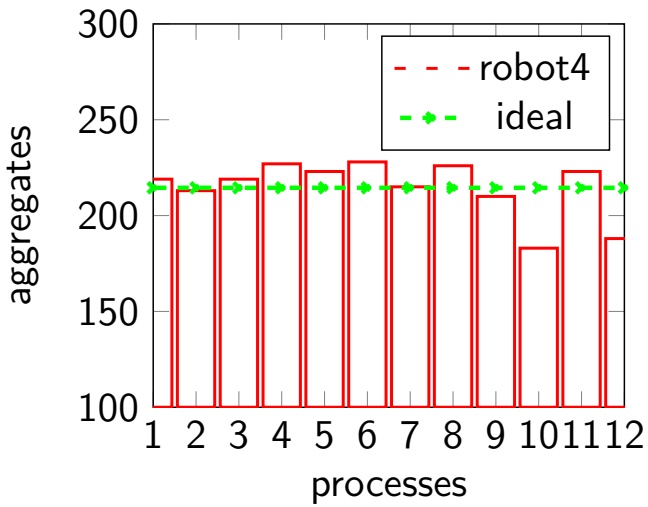


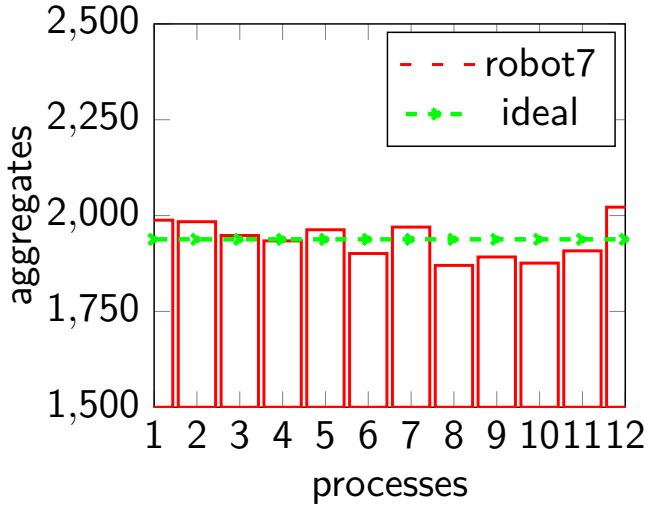
- Reduce the set of states associated with each aggrgate.
- Determining a uniquely representative per each SCC source.

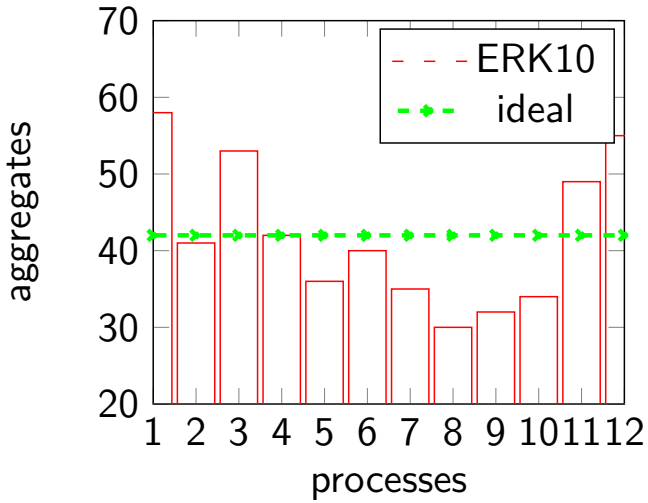
Impact of the canonicalization algorithm

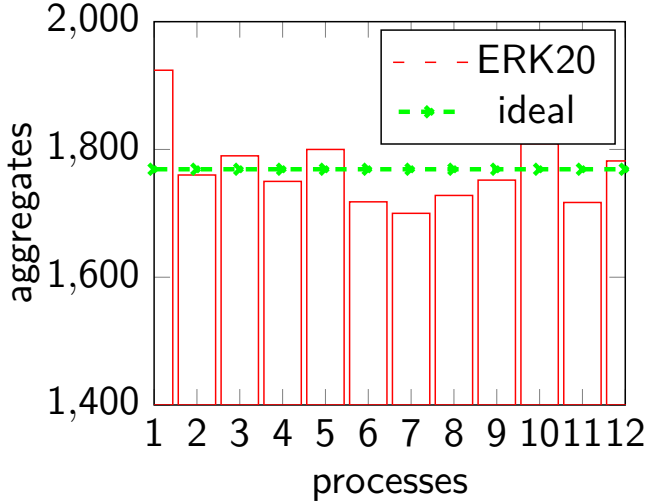
Model		Distibuted		Hybrid		Multi-thread	
Net	Seq	runtime	SpMax	runtime	SpMax	runtime	SpMax
ring4	4.56	0.77	5.9	0.38	12.0	0.84	5.4
ring5	94.42	10.74	8.8	2.96	35.6	15.49	6.1
ring6	2100.41	226.06	9.3	77.55	28.8	429.58	4.9
philo8	23.32	3.92	5.9	1.27	18.3	4.55	5.1
philo10	586.03	75.82	7.7	15.82	37.0	97.07	6.0
fms4	559,80	103.10	5.4	46.27	12.1	159.93	3.5
fms5	7433.16	168.79	6.3	315.16	23.6	1270.56	5.8
robot4	14.60	2.72	5.3	0.68	21.4	2.00	7.3
robot5	80.80	12.53	6.4	3.21	25.1	9.66	8.4
robot6	312.74	49.05	6.3	7.72	40.5	38.97	8.0
robot7	1005.91	166.99	6.0	22.14	45.4	136.96	7.3
train2	101.08	26.33	3.8	14.19	7.1	26.21	3.8
train2	82.59	14.63	5.6	6.64	12.4	12.57	6.5
train2	41.64	7.18	5.8	1.94	23.6	5.53	7.5
ERK10	264.80	34.02	7.8	12.10	21.9	27.43	9.6
ERK20	5978.54	657.47	9.1	94.86	63.0	563.82	10.6

Load Balancing









Outline

Conclusion

- We proposed a hybrid (MPI-thread) approach[a] for the construction of the Symbolic Observation Graph.
- We performed experiments and we evaluated the presented approach on a benchmark of well-known parameterized problems.
- The experimental results are encouraging, and confirm that the hybrid algorithm scales well while utilizing the available computational power to its full extent.

[a] https://depot.lipn.univ-paris13.fr/PMC-SOG/hybrid-sog

Conclusion

- We proposed a hybrid (MPI-thread) approach[a] for the construction of the Symbolic Observation Graph.
- We performed experiments and we evaluated the presented approach on a benchmark of well-known parameterized problems.
- The experimental results are encouraging, and confirm that the hybrid algorithm scales well while utilizing the available computational power to its full extent.

[a] https://depot.lipn.univ-paris13.fr/PMC-SOG/hybrid-sog

Conclusion

- We proposed a hybrid (MPI-thread) approach[a] for the construction of the Symbolic Observation Graph.
- We performed experiments and we evaluated the presented approach on a benchmark of well-known parameterized problems.
- The experimental results are encouraging, and confirm that the hybrid algorithm scales well while utilizing the available computational power to its full extent.

[a] https://depot.lipn.univ-paris13.fr/PMC-SOG/hybrid-sog

Perspectives

- More intensive evaluation on more realistic examples to confirm our interpretation of the preliminary results.
- Currently, we design a parallel state and event-based LTL model checking algorithms built on our parallel construction of the SOG.
- We are doing the implementation of the parallel-SOG based model checking using Spot[a] an object-oriented model checking library written in C++.

[a] https://spot.lrde.epita.fr/

Perspectives

- More intensive evaluation on more realistic examples to confirm our interpretation of the preliminary results.
- Currently, we design a parallel state and event-based LTL model checking algorithms built on our parallel construction of the SOG.
- We are doing the implementation of the parallel-SOG based model checking using Spot[a] an object-oriented model checking library written in C++.

[a] https://spot.lrde.epita.fr/

Perspectives

- More intensive evaluation on more realistic examples to confirm our interpretation of the preliminary results.
- Currently, we design a parallel state and event-based LTL model checking algorithms built on our parallel construction of the SOG.
- We are doing the implementation of the parallel-SOG based model checking using Spot[a] an object-oriented model checking library written in C++.

[a] https://spot.lrde.epita.fr/


thank you