Automatic Repair and Deadlock Detection for Parameterized Systems

Swen Jacobs [1] **Mouhammad Sakr** [2] Marcus Völp [2]

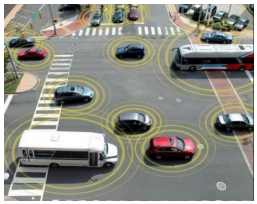
[1]CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

[2]SnT, University of Luxembourg, Luxembourg

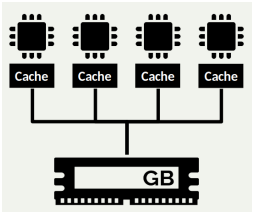
October 19, 2022

Parameterized Concurrent Systems are everywhere

source: sheffield.ac.uk



source: mtu.edu

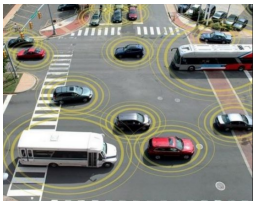


source: Meratnia et al.

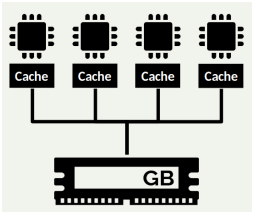
Concurrent and Parameterized Systems

Parameterized Concurrent Systems are everywhere

source: sheffield.ac.uk



source: mtu.edu



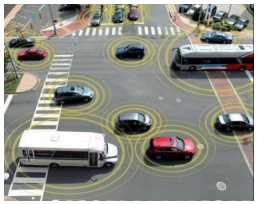
source: Meratnia et al.

Hard to get right...

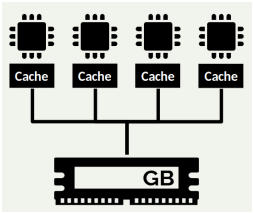
Concurrent and Parameterized Systems

Parameterized Concurrent Systems are everywhere

source: sheffield.ac.uk



source: mtu.edu



source: Meratnia et al.

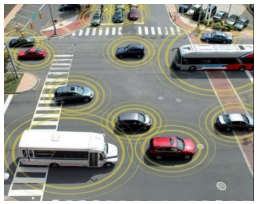
Hard to get right...

The parameterized model checking problem (PMCP) is to decide whether a temporal logic property is true for every size instance of a given system.

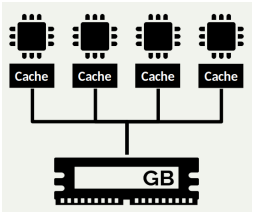
undecidable

Parameterized Concurrent Systems are everywhere

source: sheffield.ac.uk



source: mtu.edu



source: Meratnia et al.

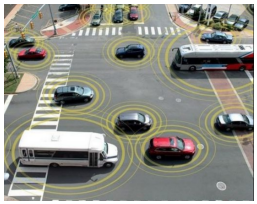
Hard to get right...

The parameterized model checking problem (PMCP) is to decide whether a temporal logic property is true for every size instance of a given system.

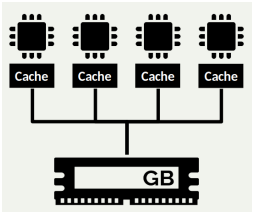
undecidable

Parameterized Concurrent Systems are everywhere

source: sheffield.ac.uk



source: mtu.edu



source: Meratnia et al.

Hard to get right...

- Homogeneous processes and have finite state space.

The parameterized model checking problem (PMCP) is to decide whether a temporal logic property is true for every size instance of a given system.

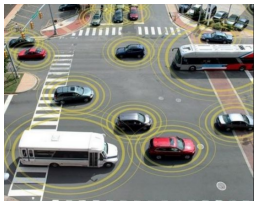
Concurrent and Parameterized Systems

decidable classes

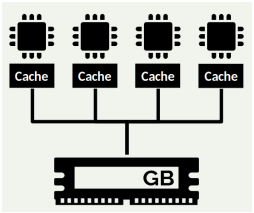
undecidable

Parameterized Concurrent Systems are everywhere

source: sheffield.ac.uk



source: mtu.edu



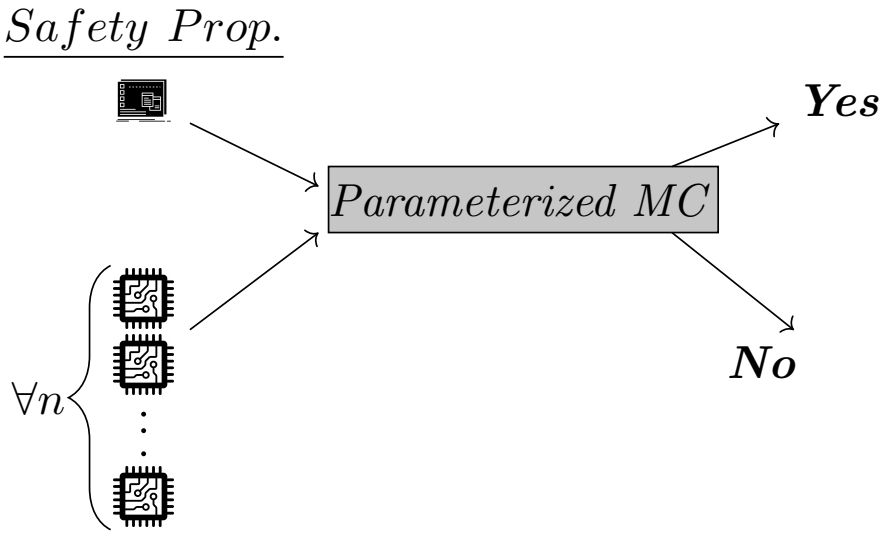
source: Meratnia et al.

Hard to get right...

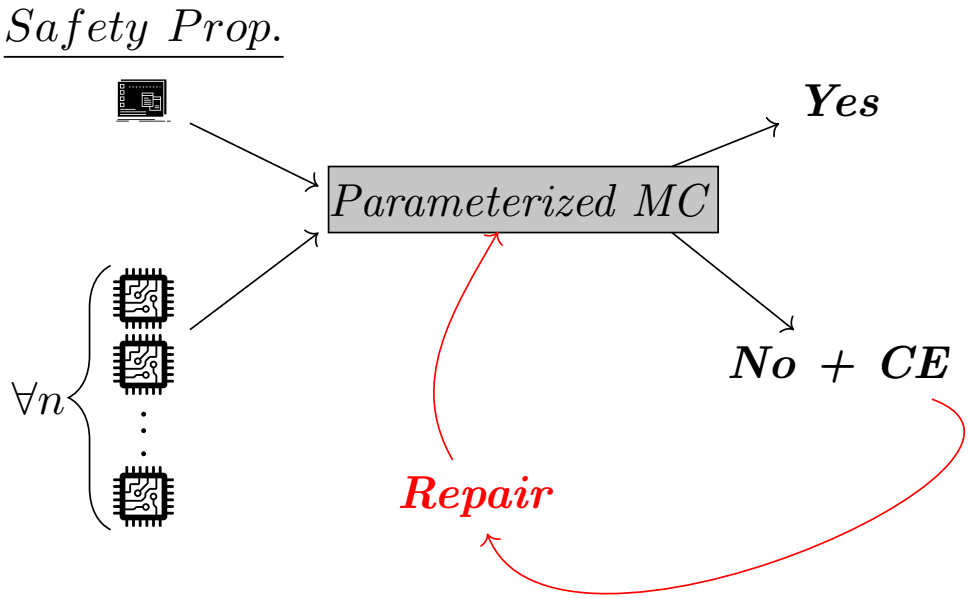
- Homogeneous processes and have finite state space.

The parameterized model checking problem (PMCP) is to decide whether a temporal logic property is true for every size instance of a given system.

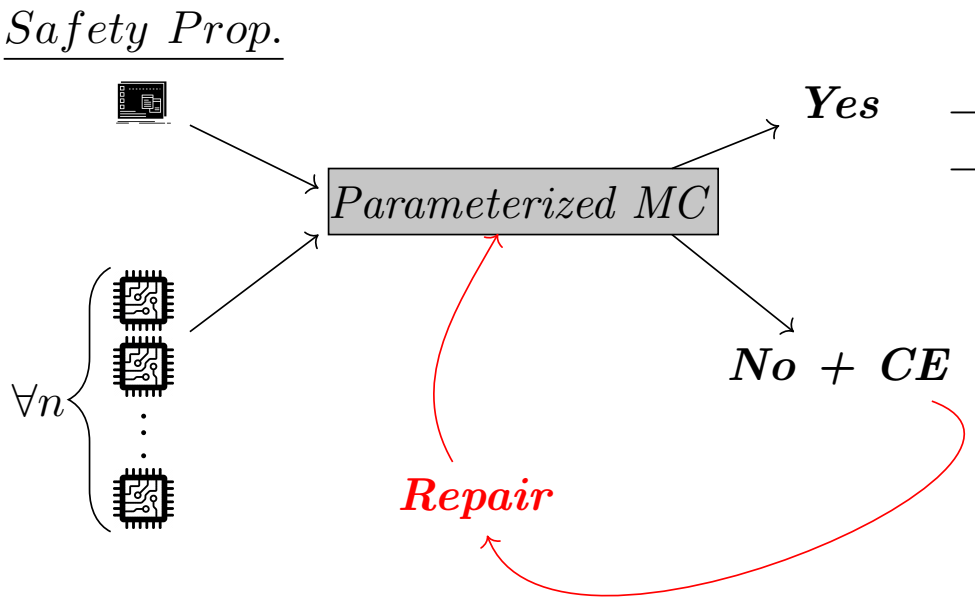
Motivation: Model Checking and Automatic Repair



Motivation: Model Checking and Automatic Repair

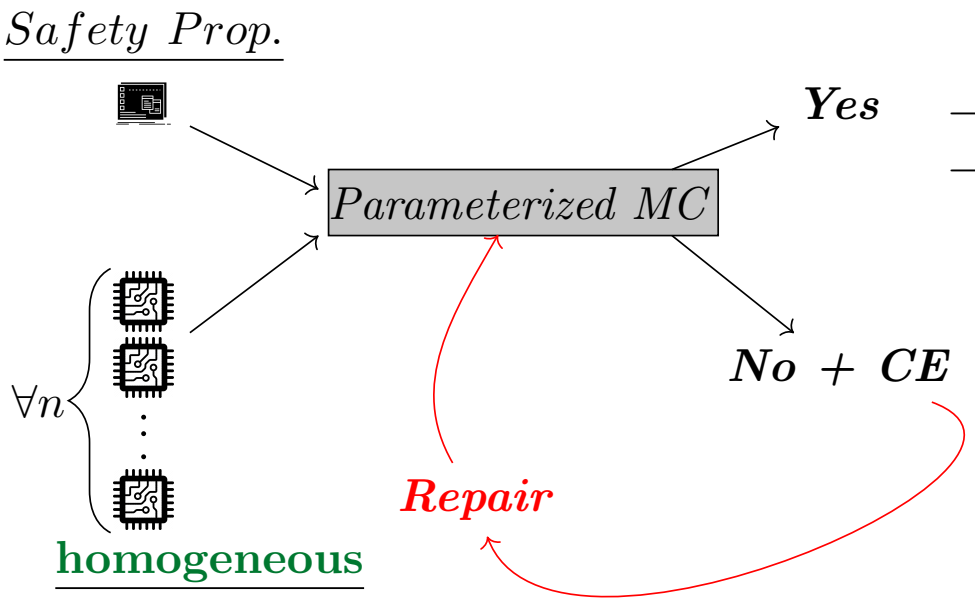


Motivation: Model Checking and Automatic Repair



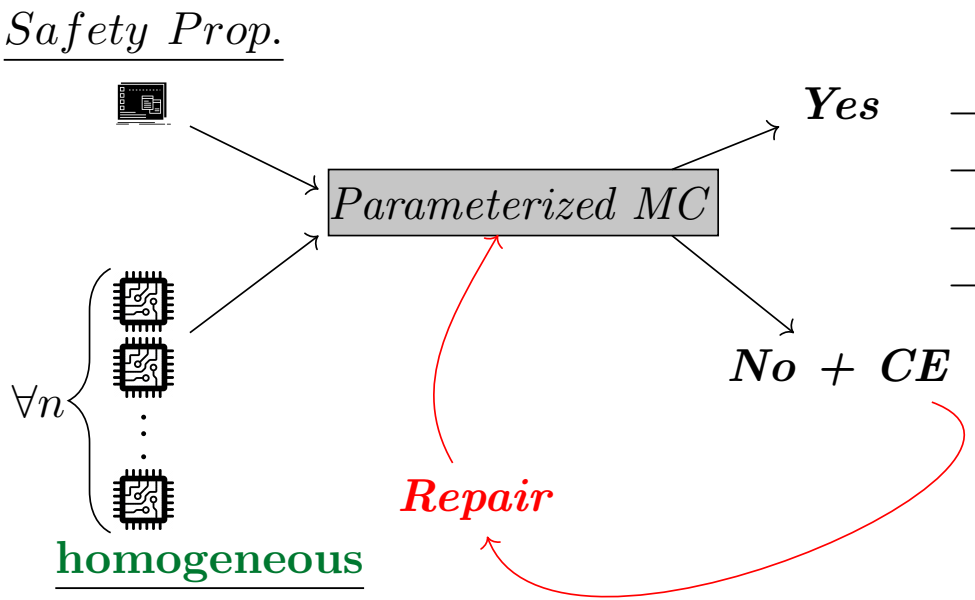
→ Repair Internal Behavior
→ Repair Communication

Motivation: Model Checking and Automatic Repair



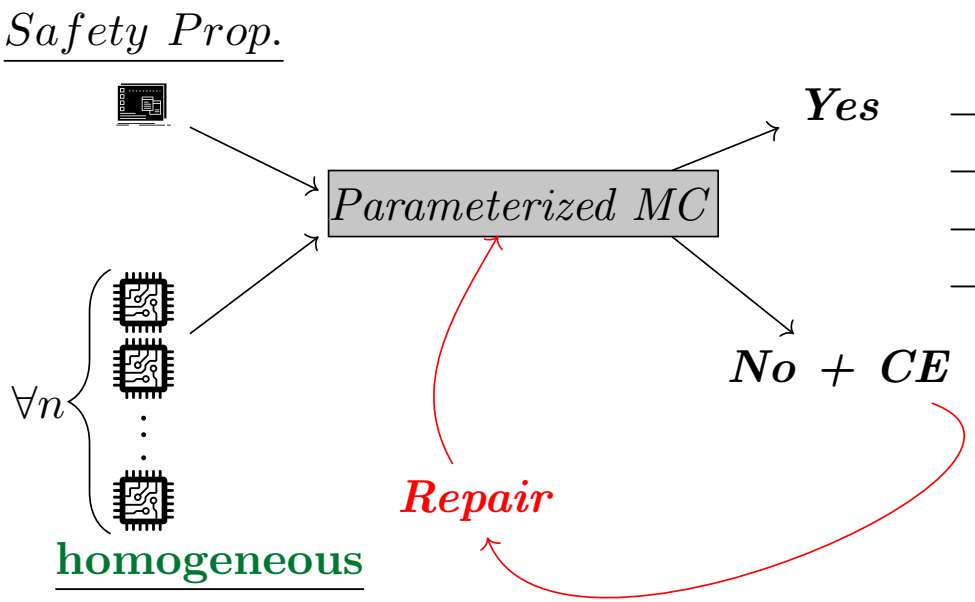
→ Repair Internal Behavior
→ Repair Communication

Motivation: Model Checking and Automatic Repair



→ Repair Internal Behavior
→ Repair Communication
→ **Avoid trivial solutions**
→ **Repair ∀n**

Motivation: Model Checking and Automatic Repair



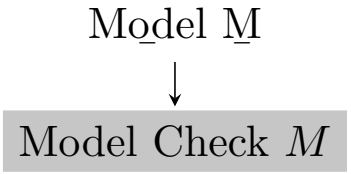
→ Repair Internal Behavior
→ Repair Communication
→ **Avoid trivial solutions**
→ **Repair ∀n**

The repair problem is, for a given process implementation, to find a **refinement** such that a given safety property is satisfied.

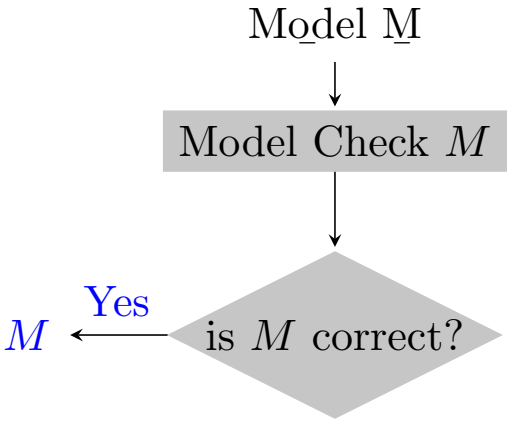
Parameterized Repair Basic Idea

Model M

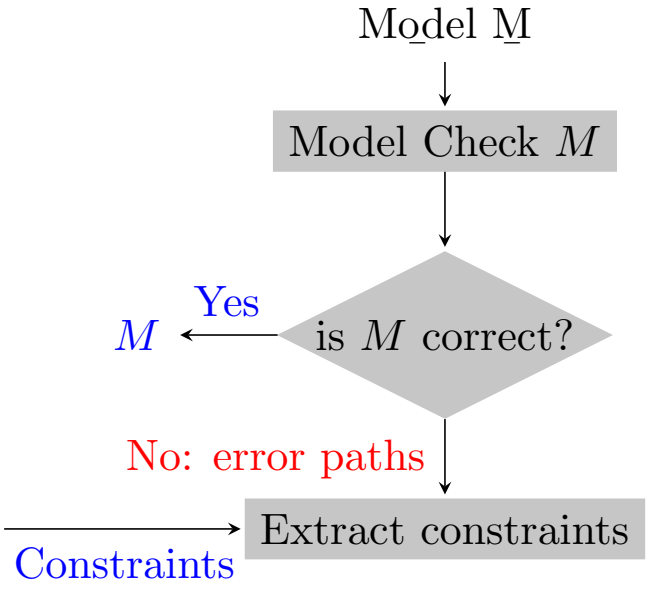
Parameterized Repair Basic Idea



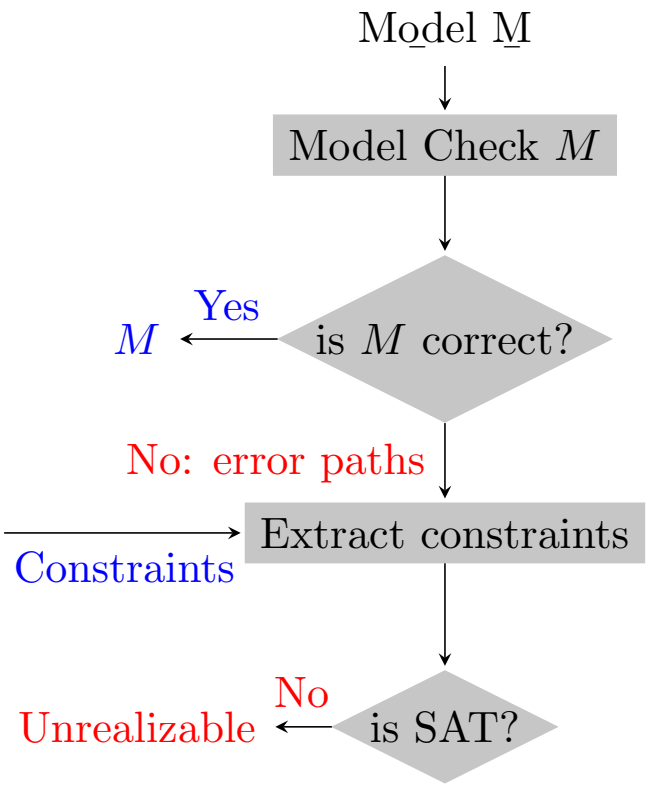
Parameterized Repair Basic Idea



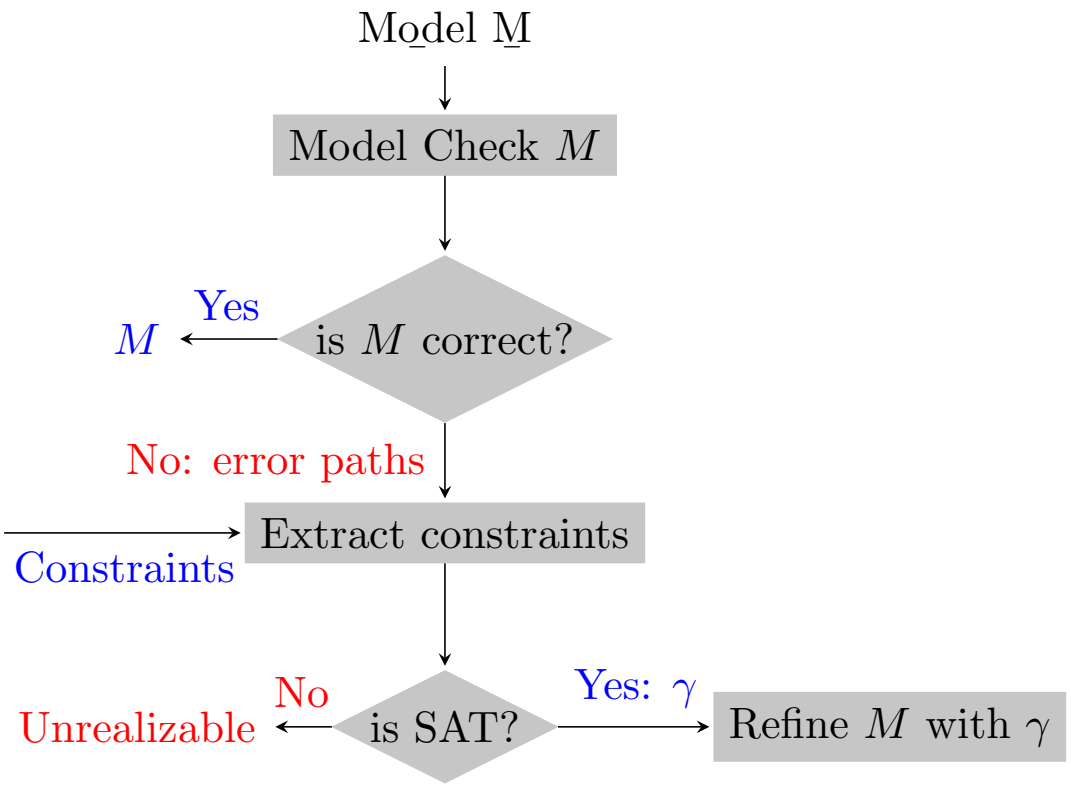
Parameterized Repair Basic Idea



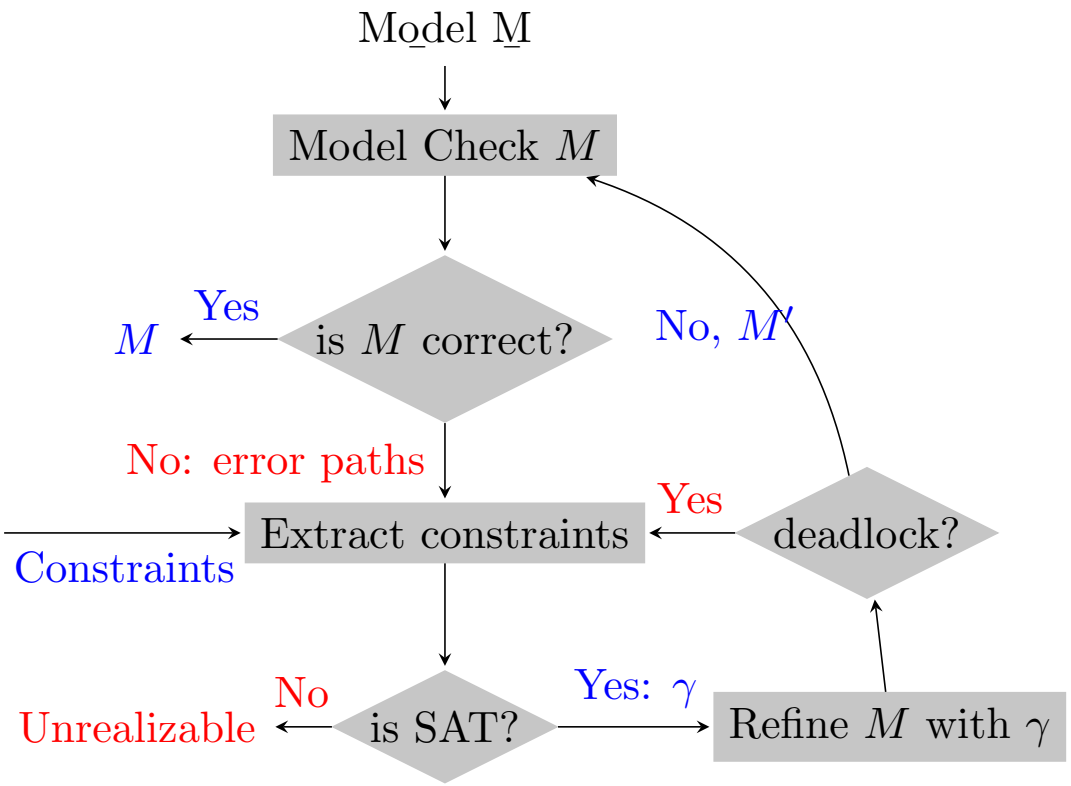
Parameterized Repair Basic Idea



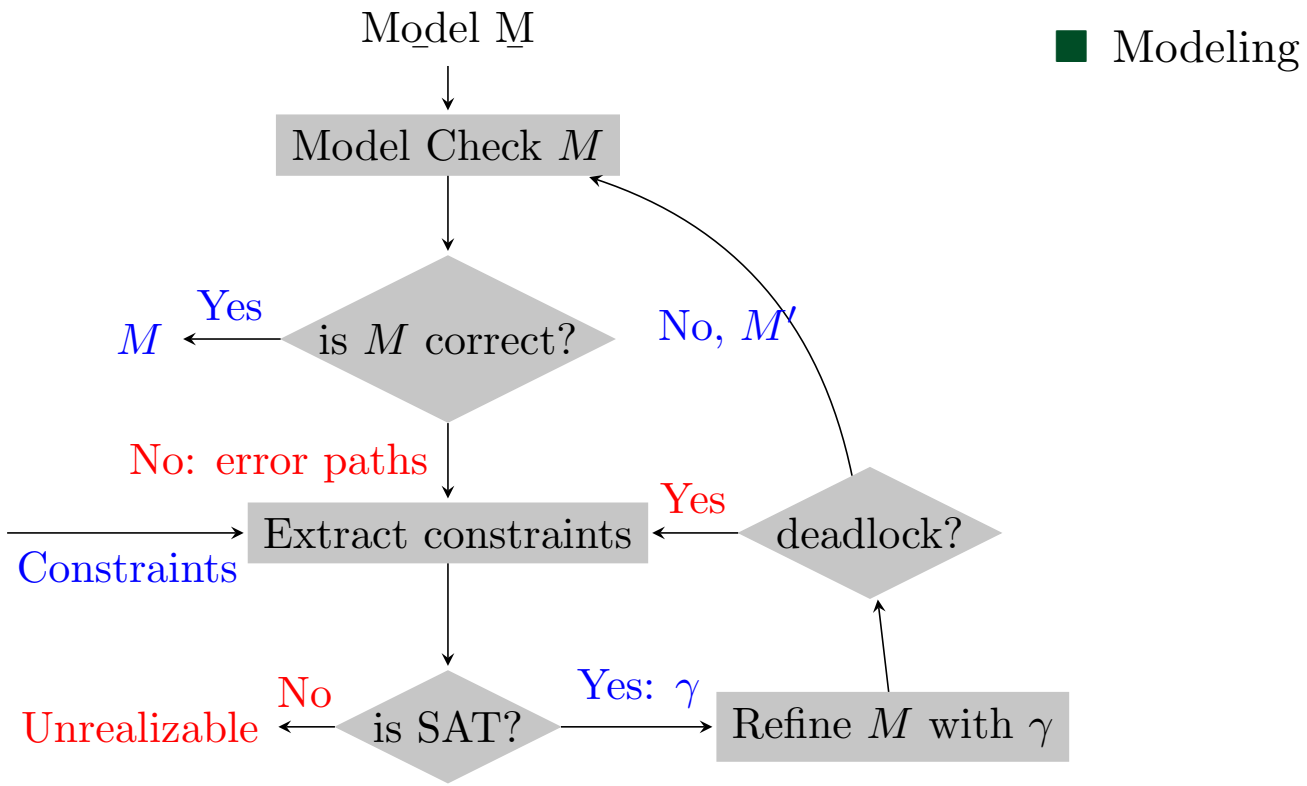
Parameterized Repair Basic Idea



Parameterized Repair Basic Idea

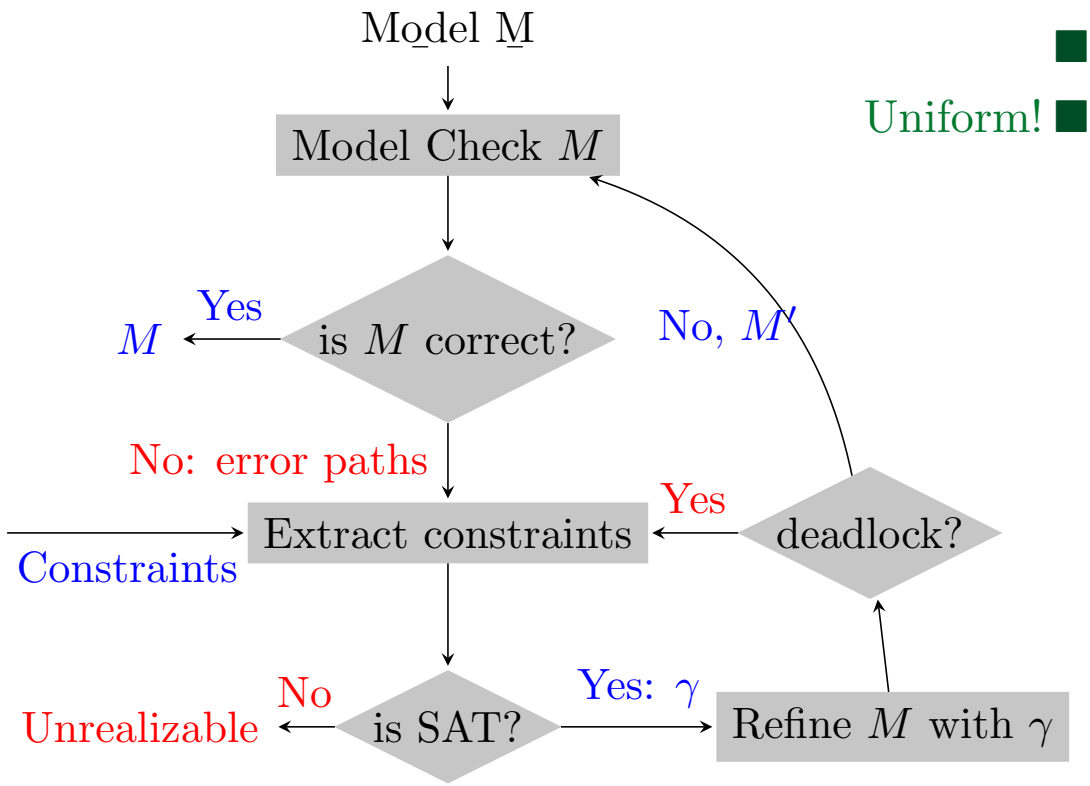


Parameterized Repair Basic Idea



Modeling

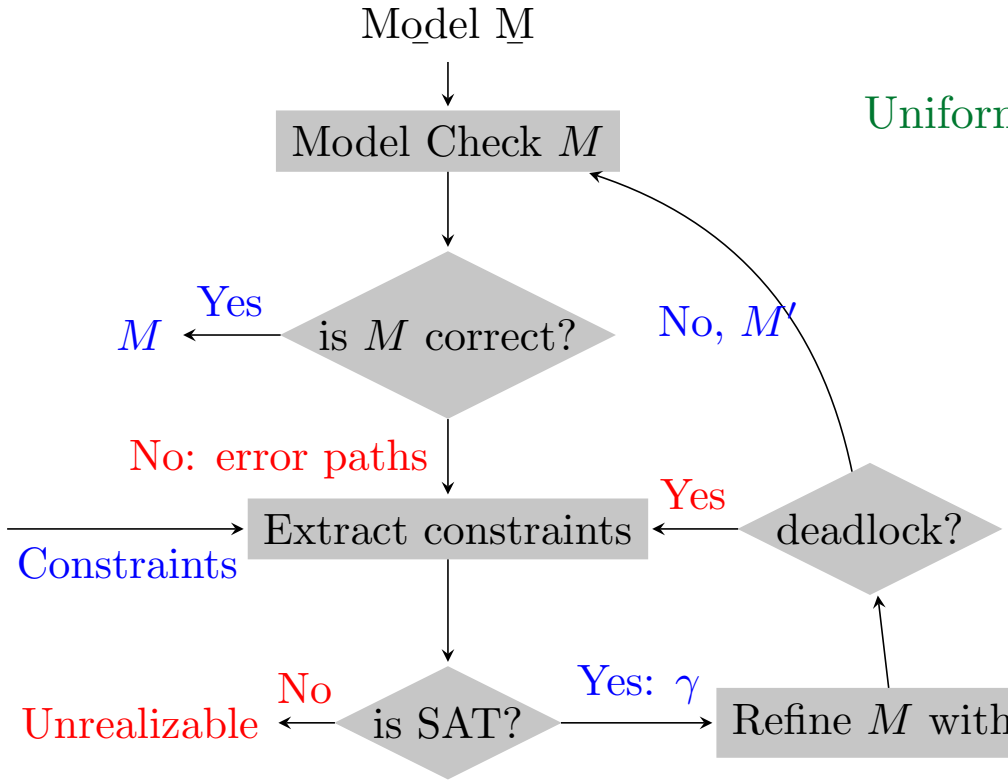
Parameterized Repair Basic Idea



Modeling

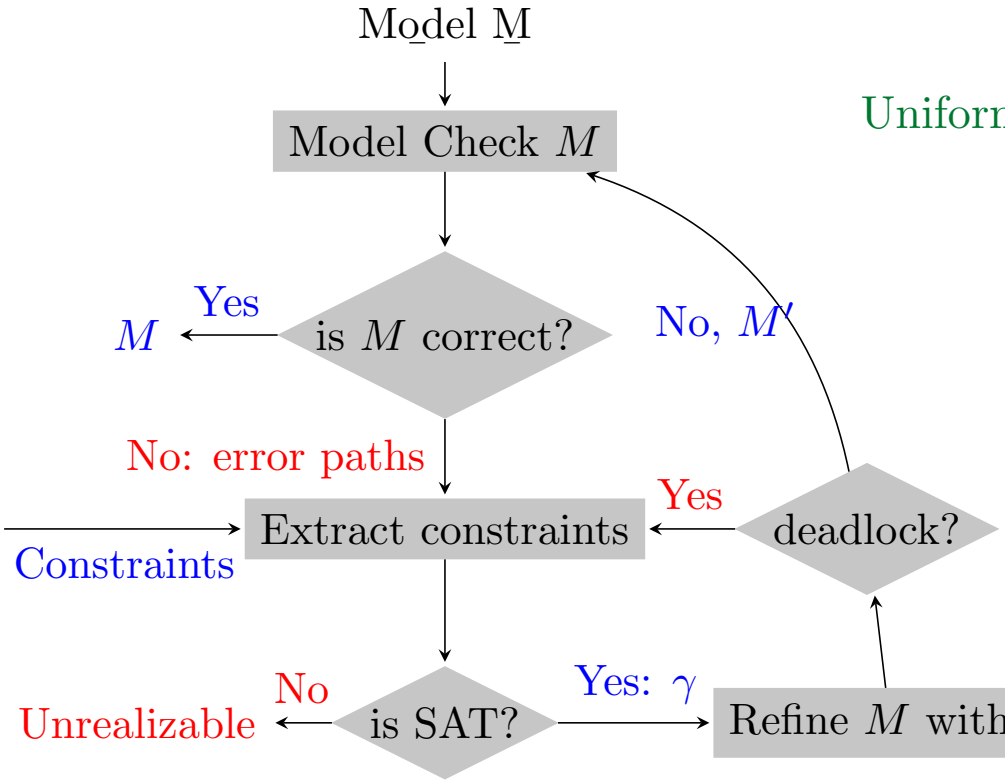
Uniform! PMC Undecidable!

Parameterized Repair Basic Idea



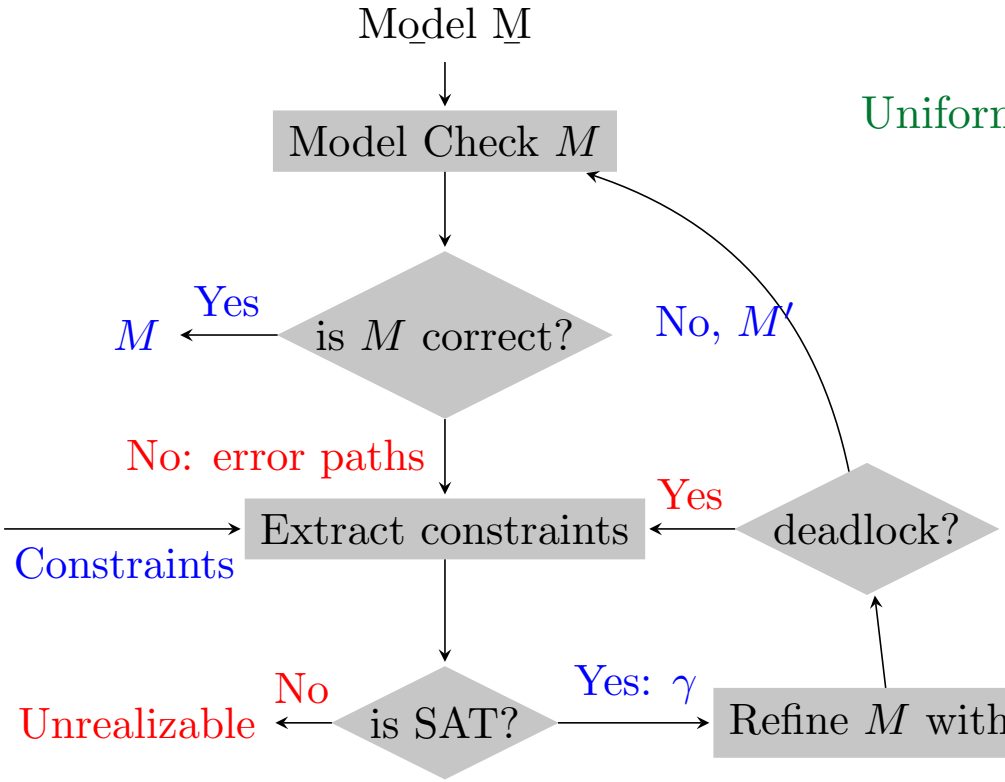
- Modeling
- PMC Uniform! Undecidable!
- Repair $\forall \mathbf{n}$

Parameterized Repair Basic Idea



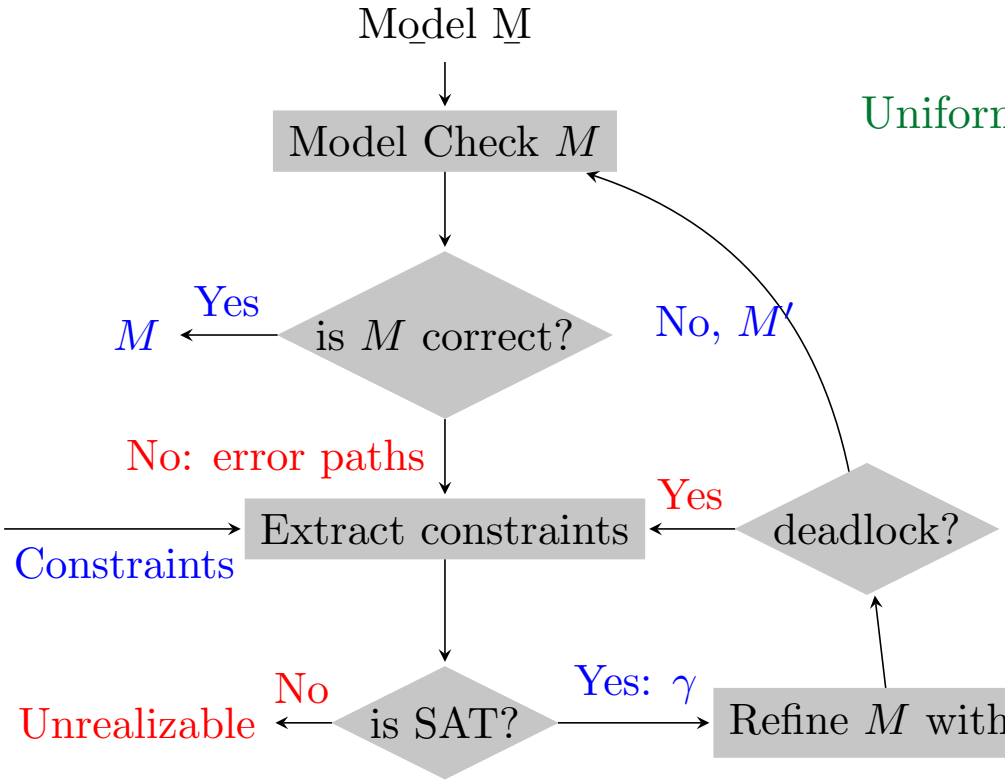
- Modeling
- Uniform! PMC Undecidable!
- Repair $\forall \mathbf{n}$
- Trivial Solutions

Parameterized Repair Basic Idea



- Modeling
- Uniform! PMC Undecidable!
- Repair $\forall \mathbf{n}$
- Trivial Solutions
- deadlock Detection Undecidable!

Parameterized Repair Basic Idea



- Modeling
- Uniform! PMC Undecidable!
- Repair $\forall \mathbf{n}$
- Trivial Solutions
- deadlock Detection Undecidable!
- Tackle Challenges
- Extensions
- Experiments

Modeling: Counter Representation

Counter System

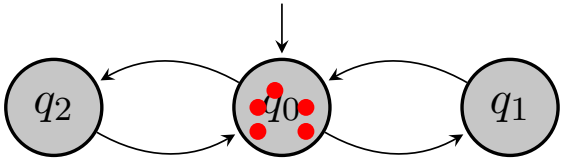
Idea: keep track of **how many processes** are in each local state.

Modeling: Counter Representation

Counter System

Idea: keep track of **how many processes** are in each local state.

$(q_0, q_0, q_0, q_0, q_0) \rightarrow (5, 0, 0)$



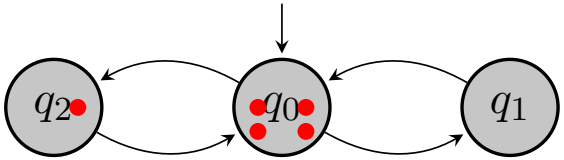
Modeling: Counter Representation

Counter System

Idea: keep track of **how many processes** are in each local state.

$(q_0, q_0, q_0, q_0, q_0) \to (5, 0, 0)$

$(q_0, q_0, q_0, q_0, q_2) \to (4, 0, 1)$



Modeling: Counter Representation

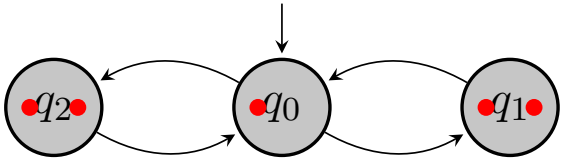
Counter System

Idea: keep track of **how many processes** are in each local state.

$(q_0, q_0, q_0, q_0, q_0) \to (5, 0, 0)$

$(q_0, q_0, q_0, q_0, q_2) \to (4, 0, 1)$

$(q_1, q_0, q_2, q_2, q_1) \to (1, 2, 2)$



Modeling: Counter Representation

Counter System

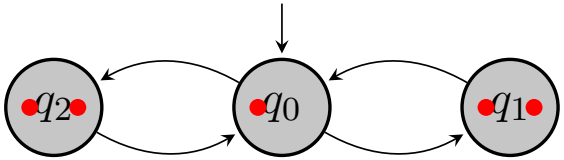
Idea: keep track of **how many processes** are in each local state.

$(q_0, q_0, q_0, q_0, q_0) \to (5, 0, 0)$

$(q_0, q_0, q_0, q_0, q_2) \to (4, 0, 1)$

$(q_1, q_0, q_2, q_2, q_1) \to (1, 2, 2)$

$q_0 \to q_1$ then $(5, 0, 0) \to (4, 1, 0)$



Modeling: Counter Representation

Counter System

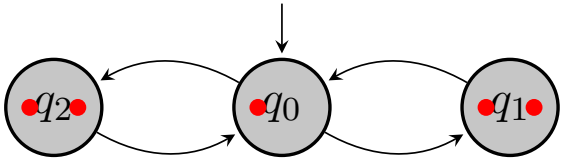
Idea: keep track of **how many processes** are in each local state.

$(q_0, q_0, q_0, q_0, q_0) \to (5, 0, 0)$

$(q_0, q_0, q_0, q_0, q_2) \to (4, 0, 1)$

$(q_1, q_0, q_2, q_2, q_1) \to (1, 2, 2)$

$q_0 \to q_1$ then $(5, 0, 0) \to (4, 1, 0)$



Infinite State

PMC: WSTS

Well Structured Transition System

$(M, \leq)$ is a **well-structured transition system** if $\leq$ is a **well quasi order** on the states of M and if $\leq$ is **compatible/monotonic** with the transition relation of M.

Well Structured Transition System

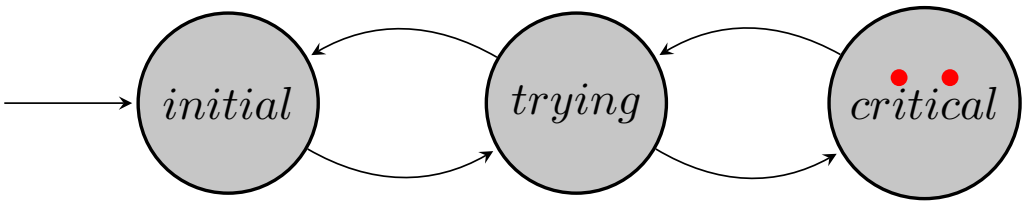
$(M, \leq)$ is a **well-structured transition system** if $\leq$ is a **well quasi order** on the states of M and if $\leq$ is **compatible/monotonic** with the transition relation of M.

- **Benefit 1:** If a set S is upward-closed then there exists a finite set B (basis) s.t. $\uparrow B = S$.

Well Structured Transition System

$(M, \leq)$ is a **well-structured transition system** if $\leq$ is a **well quasi order** on the states of M and if $\leq$ is **compatible/monotonic** with the transition relation of M.

- **Benefit 1:** If a set S is upward-closed then there exists a finite set B (basis) s.t. $\uparrow B = S$.

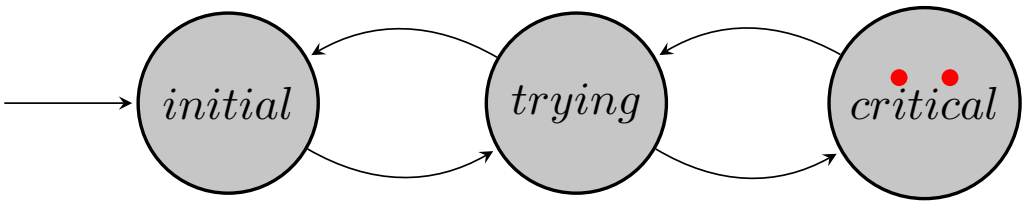


$\uparrow(0, 0, 2)$

Well Structured Transition System

$(M, \leq)$ is a **well-structured transition system** if $\leq$ is a **well quasi order** on the states of M and if $\leq$ is **compatible/monotonic** with the transition relation of M.

- **Benefit 1:** If a set S is upward-closed then there exists a finite set B (basis) s.t. $\uparrow B = S$.
- **Benefit 2:** If a set S is upward-closed then $predecessor(S)$ is upward-closed.



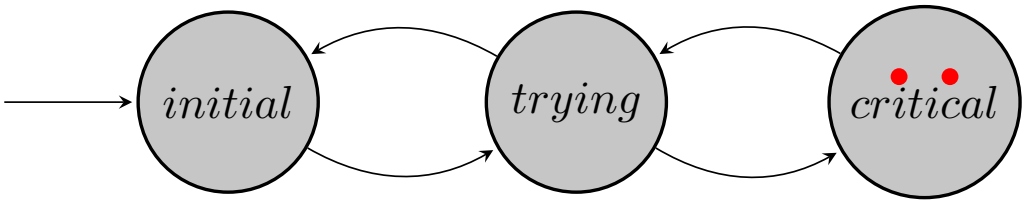
$\uparrow(0, 0, 2)$

Well Structured Transition System

$(M, \leq)$ is a **well-structured transition system** if $\leq$ is a **well quasi order** on the states of M and if $\leq$ is **compatible/monotonic** with the transition relation of M.

- **Benefit 1:** If a set S is upward-closed then there exists a finite set B (basis) s.t. $\uparrow B = S$.
- **Benefit 2:** If a set S is upward-closed then $predecessor(S)$ is upward-closed.

$(M, \leq)$ has **effective predecessor-basis** if there exists an algorithm that given a finite set of states S of M it returns a finite basis of $predecessor(\uparrow S)$.



$\uparrow(0, 0, 2)$

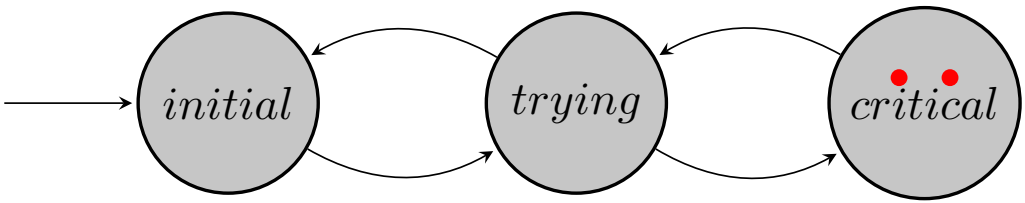
Well Structured Transition System

$(M, \leq)$ is a **well-structured transition system** if $\leq$ is a **well quasi order** on the states of M and if $\leq$ is **compatible/monotonic** with the transition relation of M.

- **Benefit 1:** If a set S is upward-closed then there exists a finite set B (basis) s.t. $\uparrow B = S$.
- **Benefit 2:** If a set S is upward-closed then $predecessor(S)$ is upward-closed.

$(M, \leq)$ has **effective predecessor-basis** if there exists an algorithm that given a finite set of states S of M it returns a finite basis of $predecessor(\uparrow S)$.

- **Benefit: Reachability analysis is decidable.**



$\uparrow(0, 0, 2)$

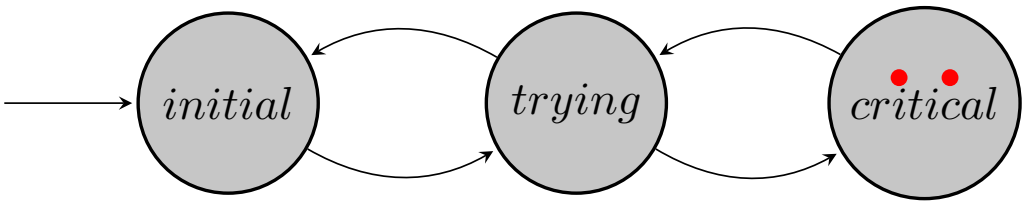
Well Structured Transition System

Given the basis B' of an infinite set E^∞, we can compute the basis B' of $predecessor(E^\infty)$

- **Benefit 1:** If a set S is upward-closed then there exists a finite set B (basis) s.t. $\uparrow B = S$.
- **Benefit 2:** If a set S is upward-closed then $predecessor(S)$ is upward-closed.

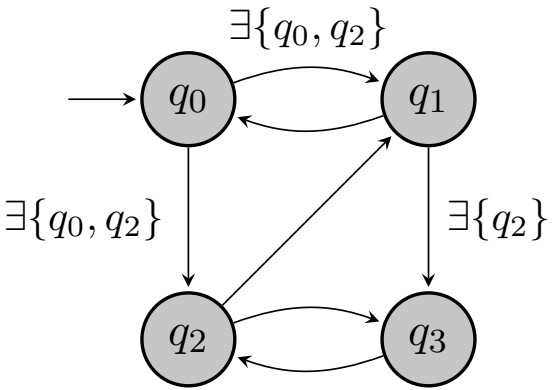
$(M, \leq)$ has **effective predecessor-basis** if there exists an algorithm that given a finite set of states S of M it returns a finite basis of $predecessor(\uparrow S)$.

- **Benefit: Reachability analysis is decidable.**

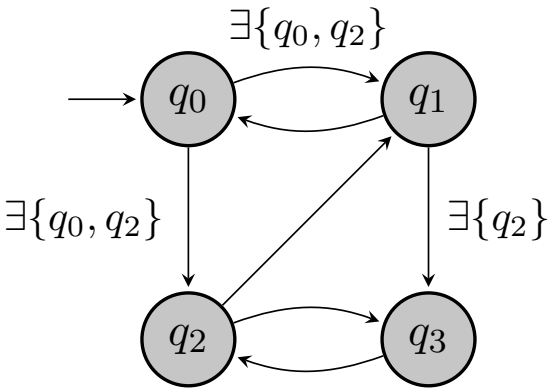


$\uparrow(0, 0, 2)$

PMC and Deadlock Detection: Disjunctive Systems



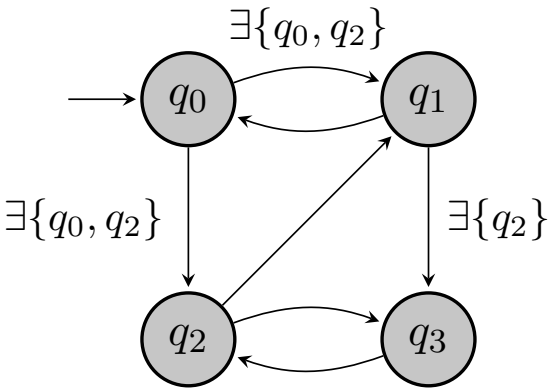
PMC and Deadlock Detection: Disjunctive Systems



Theorem

The counter representation of disjunctive systems is a ***WSTS with effective predecessor*** *basis.*

PMC and Deadlock Detection: Disjunctive Systems



Theorem

The counter representation of disjunctive systems is a ***WSTS with effective predecessor*** *basis.*

Theorem

*The counter representation M of disjunctive systems has a deadlocked run iff The 01-counter system of M has a deadlocked run (****Now****: $\underline{EXPTIME}$, ****Previously****: cutoff $= 2|Q|$).*

No Trivial Solutions: Constraint Solving

The **generation of candidate repairs** is **guided by constraints** over the local transitions.

No Trivial Solutions: Constraint Solving

The **generation of candidate repairs** is **guided by constraints** over the local transitions.

- Constraints that ensure all **error paths** discovered so far will be avoided.

No Trivial Solutions: Constraint Solving

The **generation of candidate repairs** is **guided by constraints** over the local transitions.

- Constraints that ensure all **error paths** discovered so far will be avoided.
- Constraints that express **additional desired properties** of the system.

No Trivial Solutions: Constraint Solving

The **generation of candidate repairs** is **guided by constraints** over the local transitions.

- Constraints that ensure all **error paths** discovered so far will be avoided.
- Constraints that express **additional desired properties** of the system.
  - Constraints that avoid the construction of repairs that violate the **totality assumption** on the transition relation

No Trivial Solutions: Constraint Solving

The **generation of candidate repairs** is **guided by constraints** over the local transitions.

- Constraints that ensure all **error paths** discovered so far will be avoided.
- Constraints that express **additional desired properties** of the system.
  - Constraints that avoid the construction of repairs that violate the **totality assumption** on the transition relation
  - Constraints that ensure certain **states remain reachable**.

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $\quad isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: $\quad$ **if** $isCorrect = True$ **then return** $True$
6: $\quad newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $\quad accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\quad \delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: $\quad$ **if** $isSAT = False$ **then return** $Unrealizable$
10: $\quad M = Refine(M, \delta')$
11: $\quad$ **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: $\quad$ **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0])$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0])$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

The repair Algorithm

Algorithm Parameterized Repair Algorithm

1: **ParamRepair**(M, ERR, $InitConstr$)
2: $accCnstr \leftarrow InitConstr$, $isCorrect \leftarrow False$
3: **while** $isCorrect = False$ **do**
4: $isCorrect, [RE_0, \ldots, RE_k] \leftarrow MC(M, ERR)$
5: **if** $isCorrect = True$ **then return** $True$
6: $newConstr \leftarrow \bigwedge_{s \in RE_k} BuildConstr(s, [RE_{k-1}, \ldots, RE_0]\})$
7: $accCnstr \leftarrow newConstr \wedge accCnstr$, $dlckCstr \leftarrow T$
8: $\delta', isSAT \leftarrow SAT(accCnstr \wedge dlckCstr)$
9: **if** $isSAT = False$ **then return** $Unrealizable$
10: $M = Refine(M, \delta')$
11: **if** $Deadlock(M)$ **then** $dlckCstr \leftarrow \neg\delta' \wedge dlckCstr$ *goto* 8
12: **else** *goto* 4
13: **end while**

Safety Property, Pairwise and Broadcast Systems

Safety Property, Pairwise and Broadcast Systems

- Our Algorithm can be used for general **safety properties**, based on the **automata-theoretic** approach.

Safety Property, Pairwise and Broadcast Systems

- Our Algorithm can be used for general **safety properties**, based on the **automata-theoretic** approach.
- We extended our algorithm to repair **pairwise and broadcast systems**. Both types of systems are known to be WSTS, however

Safety Property, Pairwise and Broadcast Systems

- Our Algorithm can be used for general **safety properties**, based on the **automata-theoretic** approach.
- We extended our algorithm to repair **pairwise and broadcast systems**. Both types of systems are known to be WSTS, however
  - Deadlock detection in pairwise systems **Tower-hard**

Safety Property, Pairwise and Broadcast Systems

- Our Algorithm can be used for general **safety properties**, based on the **automata-theoretic** approach.
- We extended our algorithm to repair **pairwise and broadcast systems**. Both types of systems are known to be WSTS, however
  - Deadlock detection in pairwise systems **Tower-hard**

Theorem

Deadlock detection in broadcast protocols is undecidable.

Lemma

There is a ***polynomial-time reduction*** *from the reachability problem of* ***affine VASS*** *with broadcast matrices to the deadlock detection problem in broadcast protocols.*

Experiments

SEP: Single Error Path **EPT:** Error Paths Transitions

Benchmark	Size		Errors	[SEP=F & EPT=F]			[SEP=T & EPT=F]			[SEP=F & EPT=T]			[SEP=T & EPT=T]		
	States	Edges		#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.
RW1 (PW)	5	12	C	3	2.5	4	3	2.9	4	**2**	**1.7**	**2**	**2**	**1.7**	**2**
RW2 (PW)	15	42	C	3	3.8	14	3	4.8	14	**2**	**3.2**	**7**	7	8.4	**7**
RW3 (PW)	35	102	C	3	820.7	34	3	**7.6**	34	**2**	552.3	**17**	17	40.3	**17**
RW4 (PW)	45	132	C	TO	TO	TO	**3**	**11.8**	44	TO	TO	TO	22	99.2	**22**
DLS	10	95	P1	**1**	**0.8**	13	**1**	**0.8**	13	3	2.4	**5**	5	5.6	**5**
DLS	10	95	P2	**1**	**0.8**	13	2	1.7	13	3	2.6	**9**	7	5.5	**9**
DLS	10	95	C	**2**	4.2	13	**2**	**1.5**	13	3	3	**9**	9	8.1	**9**
RF	10	147	P1	**1**	2.5	32	**1**	**1.2**	32	TO	TO	TO	8	12.4	**13**
RF	10	147	P2	**1**	**1.2**	32	**1**	1.3	32	TO	TO	TO	8	11.3	**14**
RF	10	147	C	**1**	7.8	32	**1**	**1.4**	32	TO	TO	TO	8	12.5	**12**
SD	6	39	C	**1**	**1**	**4**	**1**	**1**	**4**	3	2.4	**4**	3	3	**4**
2OT	12	128	P1	12	18.8	26	**6**	**8.3**	26	16	73.8	**17**	16	34	**17**
2OT	12	128	P2	**1**	**1.8**	26	**1**	**1.8**	26	4	2958	**11**	8	16.5	12
2OT	12	128	C	11	17.2	Unreal.	**6**	**11.7**	Unreal.	TO	TO	TO	11	48.6	Unreal.
MESI1	4	26	C	**1**	2.4	6	**1**	**0.9**	6	2	1.8	**5**	4	3.5	**5**
MESI2	9	71	C	**1**	**1.1**	26	**1**	**1.1**	26	3	56.4	20	6	6.8	**15**
MESI3	14	116	C	**1**	109.4	46	**1**	**108.1**	46	TO	TO	TO	6	289.9	**15**

Experiments

SEP: True **EPT: False**

Benchmark	Size		Errors	[SEP=F & EPT=F]			[SEP=T & EPT=F]			[SEP=F & EPT=T]			[SEP=T & EPT=T]		
	States	Edges		#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.
RW1 (PW)	5	12	C	3	2.5	4	3	2.9	4	**2**	**1.7**	**2**	**2**	**1.7**	**2**
RW2 (PW)	15	42	C	3	3.8	14	3	4.8	14	**2**	**3.2**	**7**	7	8.4	**7**
RW3 (PW)	35	102	C	3	820.7	34	3	**7.6**	34	**2**	552.3	**17**	17	40.3	**17**
RW4 (PW)	45	132	C	TO	TO	TO	**3**	**11.8**	44	TO	TO	TO	22	99.2	**22**
DLS	10	95	P1	**1**	**0.8**	13	**1**	**0.8**	13	3	2.4	**5**	5	5.6	**5**
DLS	10	95	P2	**1**	**0.8**	13	2	1.7	13	3	2.6	**9**	7	5.5	**9**
DLS	10	95	C	**2**	4.2	13	**2**	**1.5**	13	3	3	**9**	9	8.1	**9**
RF	10	147	P1	**1**	2.5	32	**1**	**1.2**	32	TO	TO	TO	8	12.4	**13**
RF	10	147	P2	**1**	**1.2**	32	**1**	1.3	32	TO	TO	TO	8	11.3	**14**
RF	10	147	C	**1**	7.8	32	**1**	**1.4**	32	TO	TO	TO	8	12.5	**12**
SD	6	39	C	**1**	**1**	**4**	**1**	**1**	**4**	3	2.4	**4**	3	3	**4**
2OT	12	128	P1	12	18.8	26	**6**	**8.3**	26	16	73.8	**17**	16	34	**17**
2OT	12	128	P2	**1**	**1.8**	26	**1**	**1.8**	26	4	2958	**11**	8	16.5	12
2OT	12	128	C	11	17.2	Unreal.	**6**	**11.7**	Unreal.	TO	TO	TO	11	48.6	Unreal.
MESI1	4	26	C	**1**	2.4	6	**1**	**0.9**	6	2	1.8	**5**	4	3.5	**5**
MESI2	9	71	C	**1**	**1.1**	26	**1**	**1.1**	26	3	56.4	20	6	6.8	**15**
MESI3	14	116	C	**1**	109.4	46	**1**	**108.1**	46	TO	TO	TO	6	289.9	**15**

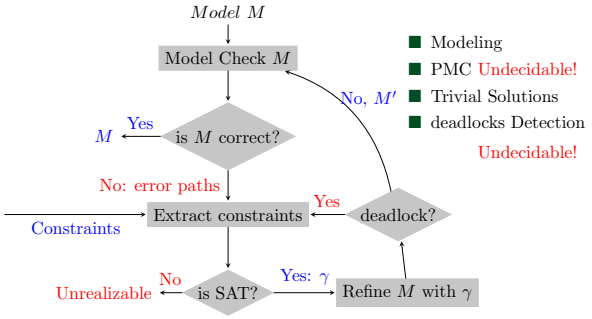
Experiments

SEP: True **EPT: True**

Benchmark	Size		Errors	[SEP=F & EPT=F]			[SEP=T & EPT=F]			[SEP=F & EPT=T]			[SEP=T & EPT=T]		
	States	Edges		#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.
RW1 (PW)	5	12	C	3	2.5	4	3	2.9	4	**2**	**1.7**	**2**	**2**	**1.7**	**2**
RW2 (PW)	15	42	C	3	3.8	14	3	4.8	14	**2**	**3.2**	**7**	7	8.4	**7**
RW3 (PW)	35	102	C	3	820.7	34	3	**7.6**	34	**2**	552.3	**17**	17	40.3	**17**
RW4 (PW)	45	132	C	TO	TO	TO	**3**	**11.8**	44	TO	TO	TO	22	99.2	**22**
DLS	10	95	P1	**1**	**0.8**	13	**1**	**0.8**	13	3	2.4	**5**	5	5.6	**5**
DLS	10	95	P2	**1**	**0.8**	13	2	1.7	13	3	2.6	**9**	7	5.5	**9**
DLS	10	95	C	**2**	4.2	13	**2**	**1.5**	13	3	3	**9**	9	8.1	**9**
RF	10	147	P1	**1**	2.5	32	**1**	**1.2**	32	TO	TO	TO	8	12.4	**13**
RF	10	147	P2	**1**	**1.2**	32	**1**	1.3	32	TO	TO	TO	8	11.3	**14**
RF	10	147	C	**1**	7.8	32	**1**	**1.4**	32	TO	TO	TO	8	12.5	**12**
SD	6	39	C	**1**	**1**	**4**	**1**	**1**	**4**	3	2.4	**4**	3	3	**4**
2OT	12	128	P1	12	18.8	26	**6**	**8.3**	26	16	73.8	**17**	16	34	**17**
2OT	12	128	P2	**1**	**1.8**	26	**1**	**1.8**	26	4	2958	**11**	8	16.5	12
2OT	12	128	C	11	17.2	Unreal.	**6**	**11.7**	Unreal.	TO	TO	TO	11	48.6	Unreal.
MESI1	4	26	C	**1**	2.4	6	**1**	**0.9**	6	2	1.8	**5**	4	3.5	**5**
MESI2	9	71	C	**1**	**1.1**	26	**1**	**1.1**	26	3	56.4	20	6	6.8	**15**
MESI3	14	116	C	**1**	109.4	46	**1**	**108.1**	46	TO	TO	TO	6	289.9	**15**

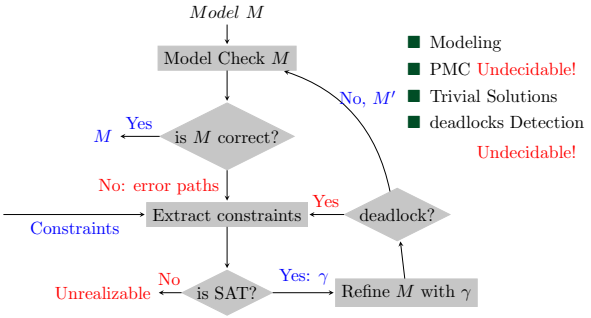
Summary

Parameterized Repair Basic Idea

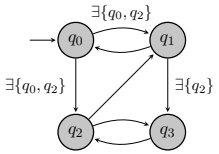


Summary

Parameterized Repair Basic Idea



Disjunctive Systems Results



Theorem

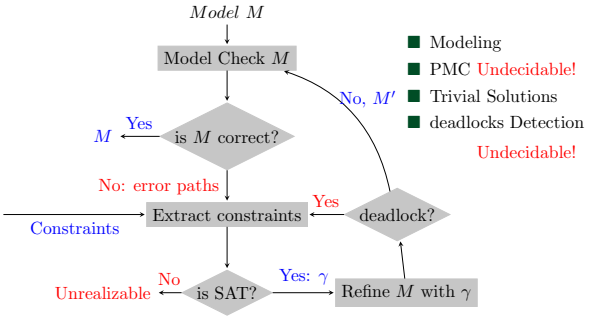
The counter representation of disjunctive systems is a WSTS with effective predecessor basis.

Theorem

*The counter representation M of disjunctive systems has a deadlocked run iff The 01-counter system of M has a deadlocked run (**Now**: $EXPTIME$, **Previously**: $Q^{2|Q|}$).*

Summary

Parameterized Repair Basic Idea



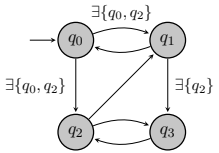
Safety Property, Pairwise and Broadcast Systems

- Our Algorithm can be used for general **safety properties**, based on the **automata-theoretic** approach.
- We extended our algorithm to repair **pairwise and broadcast systems**. Both types of systems are known to be WSTS, however
  - Deadlock detection in pairwise systems **Tower-hard**

Theorem

Deadlock detection in broadcast protocols is undecidable.

Disjunctive Systems Results



Theorem

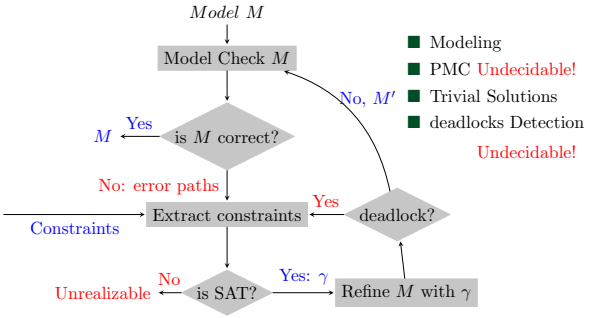
The counter representation of disjunctive systems is a WSTS with effective predecessor basis.

Theorem

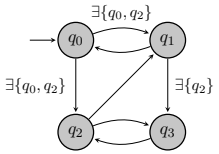
*The counter representation M of disjunctive systems has a deadlocked run iff The 01-counter system of M has a deadlocked run (**Now**: $EXPTIME$, **Previously**: $Q^{2|Q|}$).*

Summary

Parameterized Repair Basic Idea



Disjunctive Systems Results



Theorem

The counter representation of disjunctive systems is a WSTS with effective predecessor basis.

Theorem

*The counter representation M of disjunctive systems has a deadlocked run iff The 01-counter system of M has a deadlocked run (**Now**: $EXPTIME$, **Previously**: $Q^{2^{|Q|}}$).*

Safety Property, Pairwise and Broadcast Systems

- Our Algorithm can be used for general **safety properties**, based on the **automata-theoretic** approach.
- We extended our algorithm to repair **pairwise and broadcast systems**. Both types of systems are known to be WSTS, however
  - Deadlock detection in pairwise systems **Tower-hard**

Theorem

Deadlock detection in broadcast protocols is undecidable.

Experiments

SEP: Single Error Path **EPT: Error Paths Transitions**

Benchmark	Size		Errors	[SEP=F & EPT=F]			[SEP=T & EPT=F]			[SEP=F & EPT=T]			[SEP=T & EPT=T]		
	States	Edges		#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.	#Iter	Time	#D.T.
RW1 (PW)	5	12	C	3	2.5	4	3	2.9	4	**2**	**1.7**	**2**	**2**	**1.7**	**2**
RW2 (PW)	15	42	C	3	3.8	14	3	4.8	14	**2**	**3.2**	**7**	7	8.4	**7**
RW3 (PW)	35	102	C	3	820.7	34	3	**7.6**	34	**2**	552.3	**17**	17	40.3	**17**
RW4 (PW)	45	132	C	TO	TO	TO	**3**	**11.8**	44	TO	TO	TO	22	99.2	**22**
DLS	10	95	P1	**1**	**0.8**	13	**1**	**0.8**	13	3	2.4	**5**	5	5.6	**5**
DLS	10	95	P2	**1**	**0.8**	13	2	1.7	13	3	2.6	**9**	7	5.5	**9**
DLS	10	95	C	**2**	4.2	13	**2**	**1.5**	13	3	3	**9**	9	8.1	**9**
RF	10	147	P1	**1**	2.5	32	**1**	**1.2**	32	TO	TO	TO	8	12.4	**13**
RF	10	147	P2	**1**	**1.2**	32	**1**	1.3	32	TO	TO	TO	8	11.3	**14**
RF	10	147	C	**1**	7.8	32	**1**	**1.4**	32	TO	TO	TO	8	12.5	**12**
SD	6	39	C	**1**	**1**	**4**	**1**	**1**	**4**	3	2.4	**4**	3	3	**4**
2OT	12	128	P1	12	18.8	26	**6**	**8.3**	26	16	73.8	**17**	16	34	**17**
2OT	12	128	P2	**1**	**1.8**	26	**1**	**1.8**	26	4	2958	**11**	8	16.5	12
2OT	12	128	C	11	17.2	Unreal.	**6**	**11.7**	Unreal.	TO	TO	TO	11	48.6	Unreal.
MESI1	4	26	C	**1**	2.4	6	**1**	**0.9**	6	2	1.8	**5**	4	3.5	**5**
MESI2	9	71	C	**1**	**1.1**	26	**1**	**1.1**	26	3	56.4	20	6	6.8	**15**
MESI3	14	116	C	**1**	109.4	46	**1**	**108.1**	46	TO	TO	TO	6	289.9	**15**

Constraints Example

Disjunctive Systems: $A||B^n$

$$TRConstr_{Disj} = \bigwedge_{q_A \in Q_A} \bigvee_{t_A \in \delta_A(q_A)} t_A \wedge \bigwedge_{q_B \in Q_B} \bigvee_{t_B \in \delta_B(q_B)} t_B$$

Broadcast Systems: A^n

$$\bigwedge_{a \in \Sigma_{sync}} [(t_{a!!} \wedge (\bigvee_{t_{a??} \in \delta_B} t_{a??})) \vee (\neg t_{a!!} \wedge (\bigwedge_{t_{a??} \in \delta_B} \neg t_{a??}))]$$

Pairwise Systems: A^n

$$\bigwedge_{a \in \Sigma_{sync}} [(t_{a!} \wedge (\bigvee_{t_{a?} \in \delta} t_{a?})) \vee (\neg t_{a!} \wedge (\bigwedge_{t_{a?} \in \delta} \neg t_{a?}))]$$

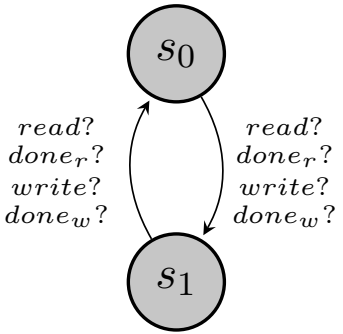
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

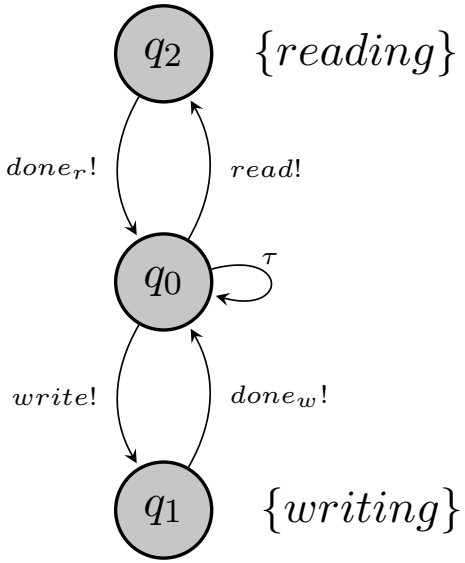
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1$, $s_0 \xrightarrow{a?} s_1$

Scheduler



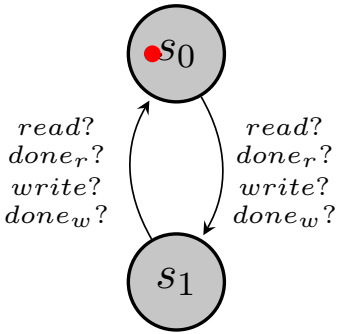
Reader-Writer



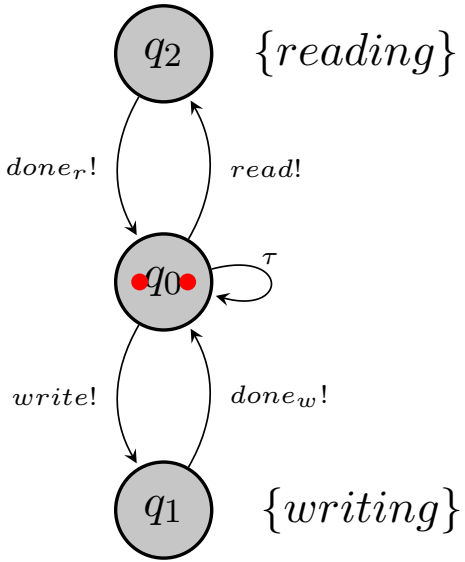
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

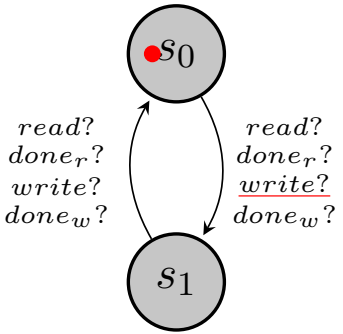


s_0, q_0, q_0

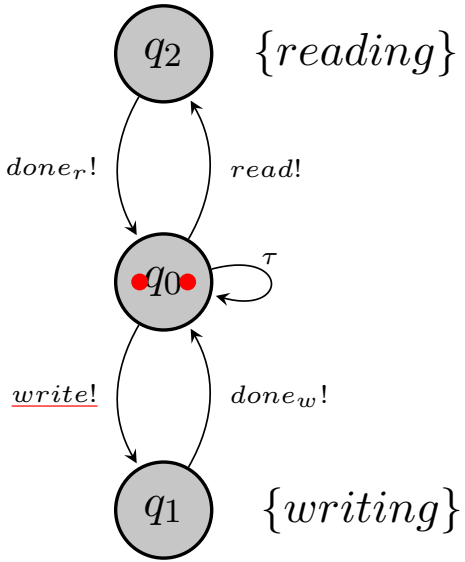
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

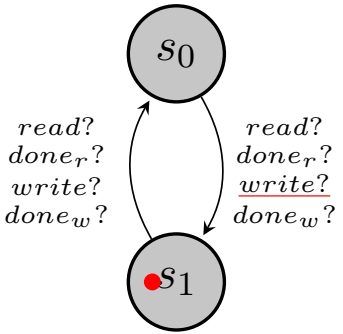


$s_0, q_0, q_0 \xrightarrow{(write!, write?)}$

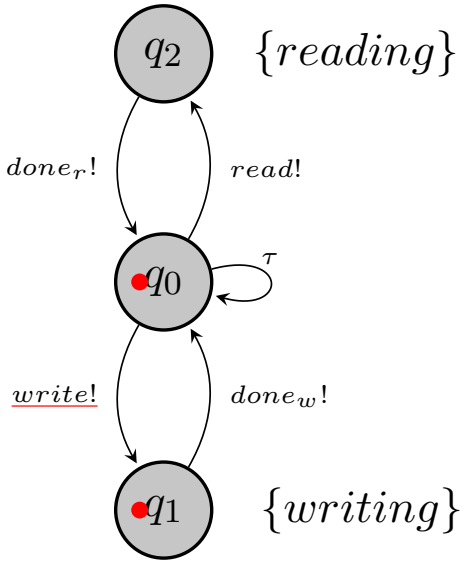
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

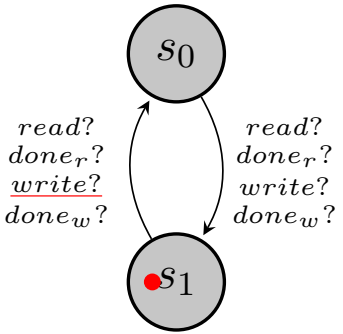


$$s_0, q_0, q_0 \xrightarrow{(write!, write?)} s_1, q_1, q_0$$

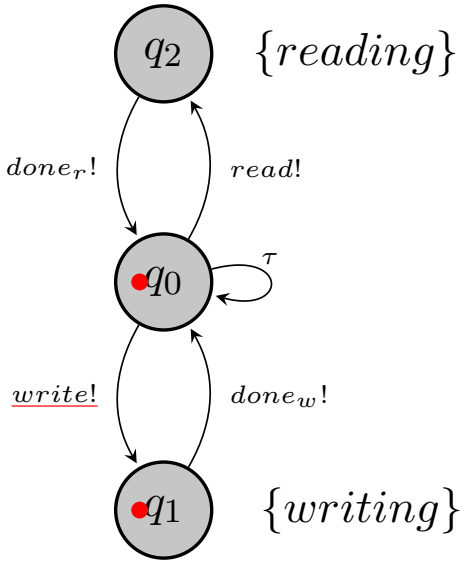
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

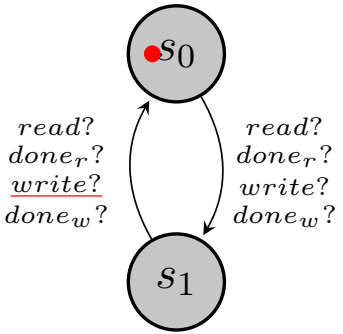


$s_0, q_0, q_0 \xrightarrow{(write!,write?)} s_1, q_1, q_0 \xrightarrow{(write!,write?)}$

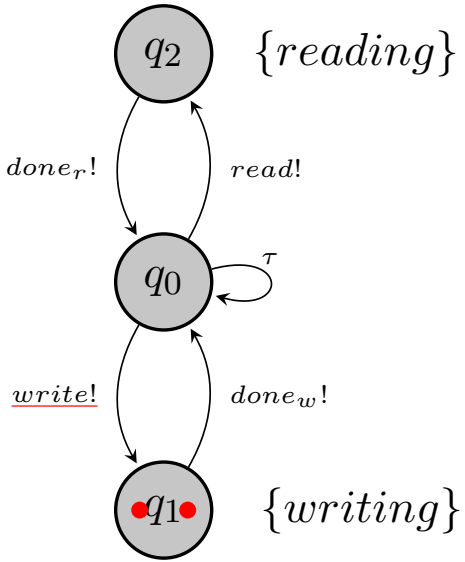
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

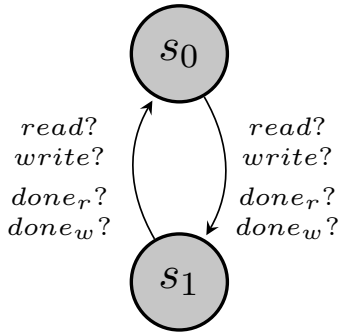


$s_0, q_0, q_0 \xrightarrow{(write!,write?)} s_1, q_1, q_0 \xrightarrow{(write!,write?)} s_0, q_1, q_1$

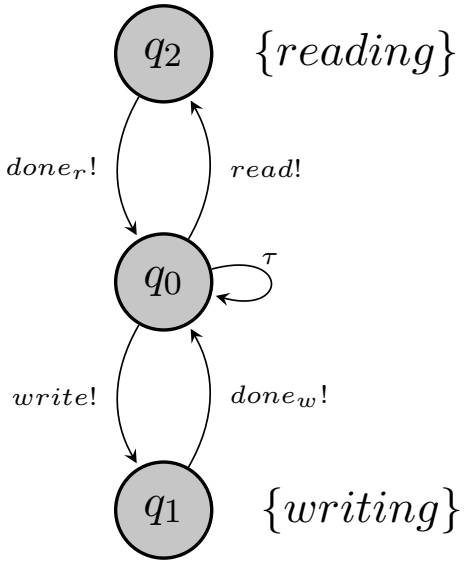
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1$, $s_0 \xrightarrow{a?} s_1$

Scheduler



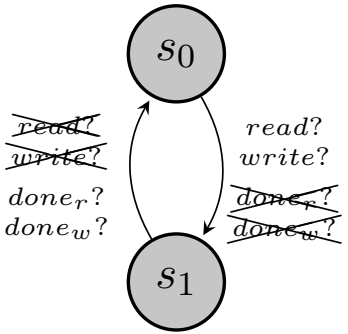
Reader-Writer



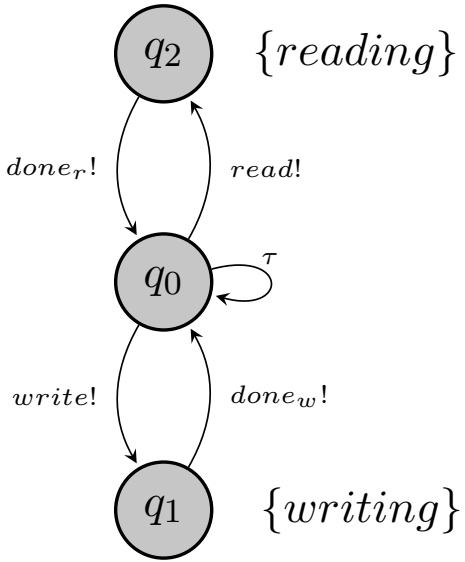
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1$, $s_0 \xrightarrow{a?} s_1$

Scheduler



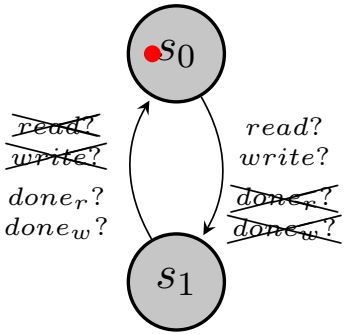
Reader-Writer



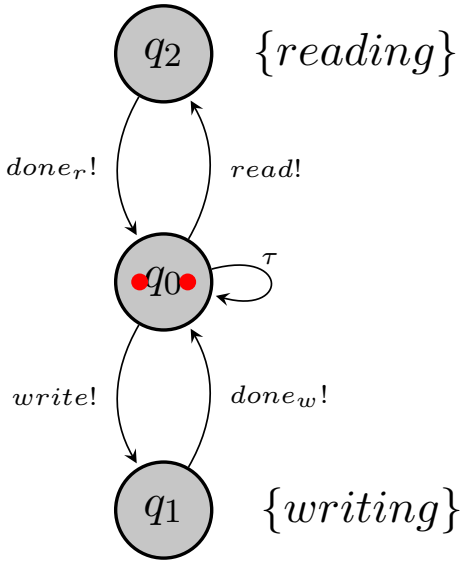
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

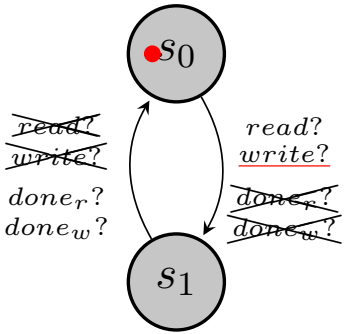


s_0, q_0, q_0

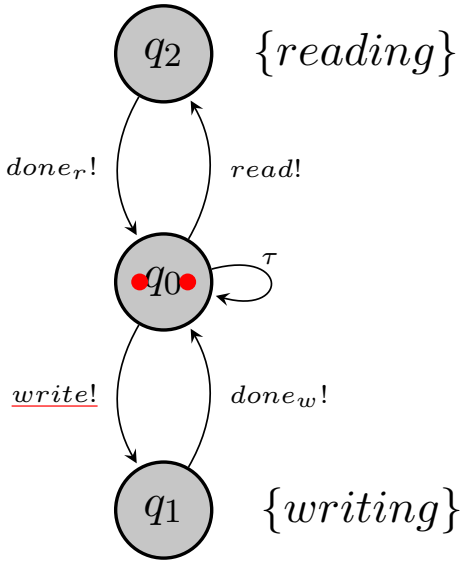
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

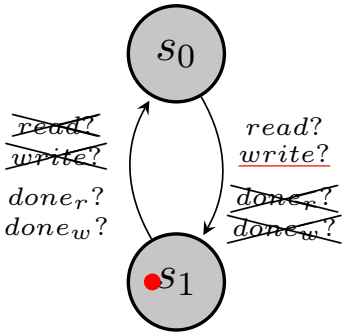


$s_0, q_0, q_0 \xrightarrow{(write!, write?)}$

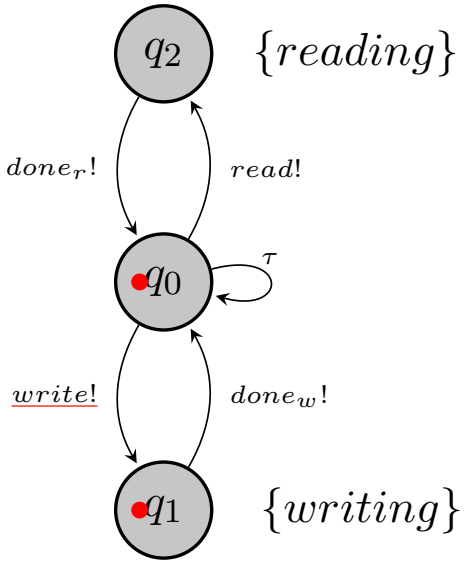
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer

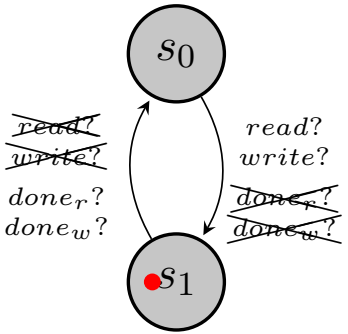


$$s_0, q_0, q_0 \xrightarrow{(write!, write?)} s_1, q_1, q_0$$

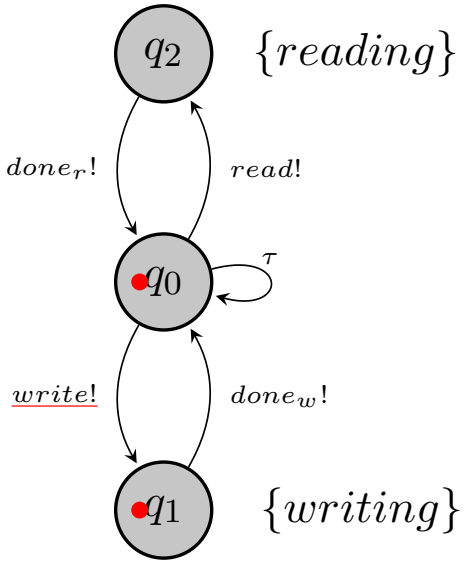
Faulty Pairwise Rendezvous System

Pairwise Synchronization: $q_0 \xrightarrow{a!} q_1,\ s_0 \xrightarrow{a?} s_1$

Scheduler



Reader-Writer



$$s_0, q_0, q_0 \xrightarrow{(write!, write?)} s_1, q_1, q_0 \xrightarrow{(write!, write?)}$$