

Extrapolation-based Path Invariants for Abstraction Refinement

A Generic CEGAR Approach for Infinite State Systems

Alexander Heußner¹ Tristan Le Gall² Grégoire Sutre¹

¹LaBRI, Université Bordeaux, CNRS, and
ANR AVeriSS

²Université Libre de Bruxelles (ULB)

SPIN Workshop Grenoble June 2009

Extrapolation-based Path Invariants for Abstraction Refinement

A Generic **CEGAR** Approach for **Infinite** State Systems

Alexander Heußner¹ Tristan Le Gall² Grégoire Sutre¹

¹LaBRI, Université Bordeaux, CNRS, and
ANR AVeriSS

²Université Libre de Bruxelles (ULB)

SPIN Workshop Grenoble June 2009

Extrapolation-based **Path Invariants** for Abstraction Refinement

A Generic CEGAR Approach for Infinite State Systems

Alexander Heußner¹ Tristan Le Gall² Grégoire Sutre¹

¹LaBRI, Université Bordeaux, CNRS, and
ANR AVeriSS

²Université Libre de Bruxelles (ULB)

SPIN Workshop Grenoble June 2009

Extrapolation-based Path Invariants for Abstraction Refinement

A Generic CEGAR Approach for Infinite State Systems

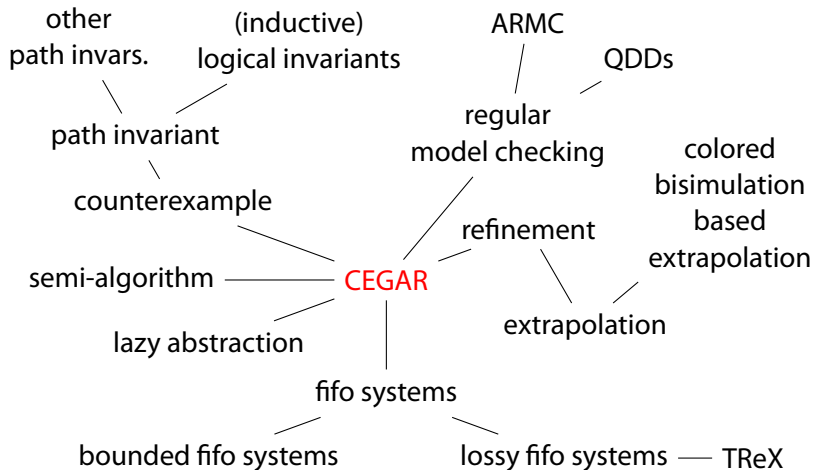
Alexander Heußner¹ Tristan Le Gall² Grégoire Sutre¹

¹LaBRI, Université Bordeaux, CNRS, and
ANR AVeriSS

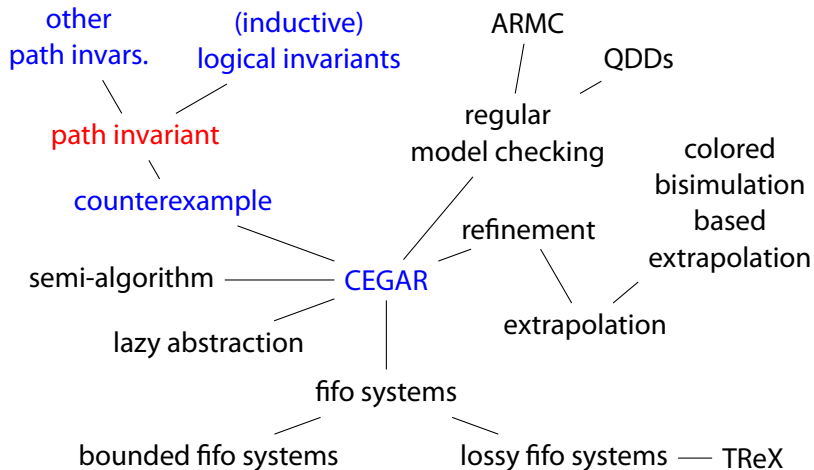
²Université Libre de Bruxelles (ULB)

SPIN Workshop Grenoble June 2009

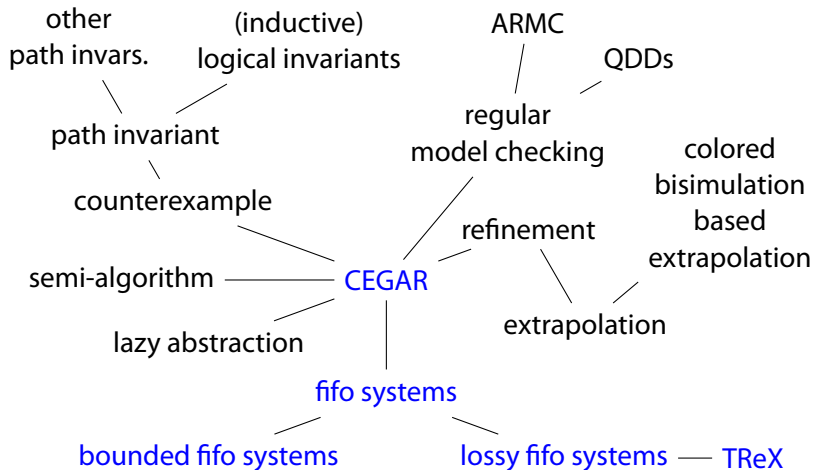
Collecting Buzzwords



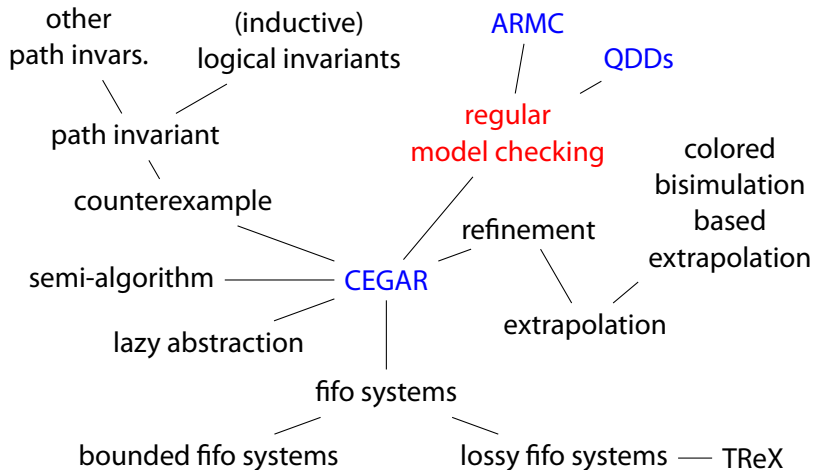
Collecting Buzzwords



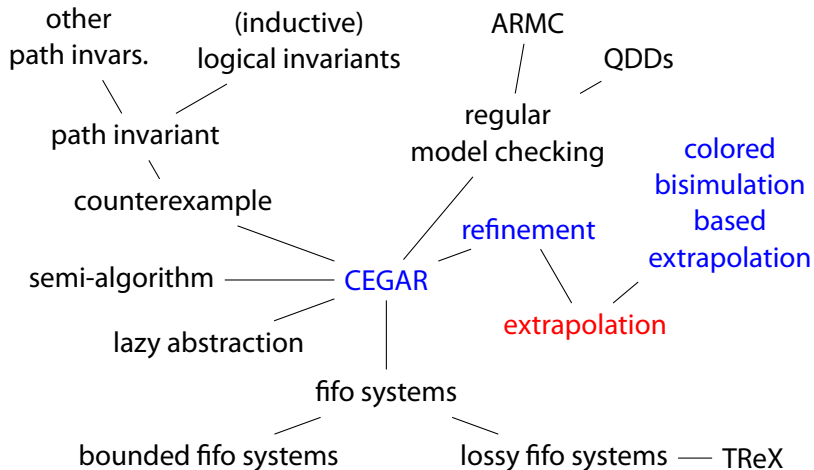
Collecting Buzzwords



Collecting Buzzwords



Collecting Buzzwords



FIFO Systems

- communicating processes / threads / peers / ...
- model each process as **finite state automaton**
- that communicates **asynchronously** via channels
 - which are **first-in first-out**,
 - **reliable** (no loss, no insertion !),
 - and **unbounded**
- these describe an **infinite** transition system
- which is Turing-powerful
 - (⚡ control state reachability is undecidable ⚡ [Brand/Zafir. '83])

FIFO Systems

- communicating processes / threads / peers / ...
- model each process as **finite state automaton**
- that communicates **asynchronously** via channels
 - which are **first-in first-out**,
 - **reliable** (no loss, no insertion !),
 - and **unbounded**
- these describe an **infinite** transition system
- which is Turing-powerful
(⚡ control state reachability is undecidable ⚡ [Brand/Zafir. '83])

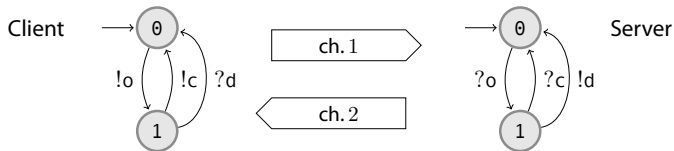
for example:
TCP-connections
as base of Berkeley
Socket API based
distributed systems

FIFO Systems

- communicating processes / threads / peers / ...
- model each process as **finite state automaton**
- that communicates **asynchronously** via channels
 - which are **first-in first-out**,
 - **reliable** (no loss, no insertion !),
 - and **unbounded**
- these describe an **infinite** transition system
- which is Turing-powerful
 - (⚡ control state reachability is undecidable ⚡ [Brand/Zafir. '83])

FIFO Systems

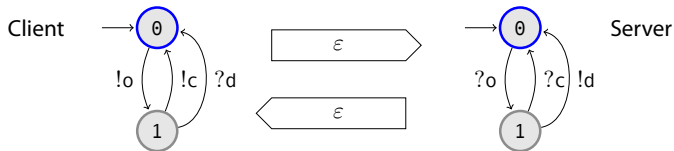
Example: Connection / Disconnection Protocol



- protocol originally "borrowed" from [Jard/Raynal '86]

FIFO Systems

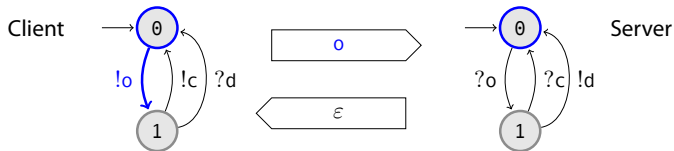
Example: Connection / Disconnection Protocol



$\langle (\theta, \theta), (\epsilon, \epsilon) \rangle$

FIFO Systems

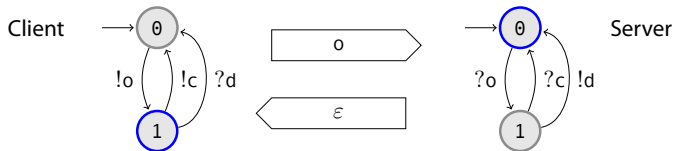
Example: Connection / Disconnection Protocol



$$\langle (\emptyset, \emptyset), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o}$$

FIFO Systems

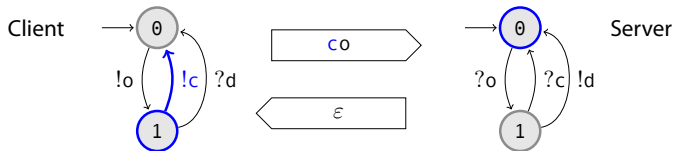
Example: Connection / Disconnection Protocol



$$\langle (\emptyset, \emptyset), (\epsilon, \epsilon) \rangle \xrightarrow{!o} \langle (1, \emptyset), (o, \epsilon) \rangle$$

FIFO Systems

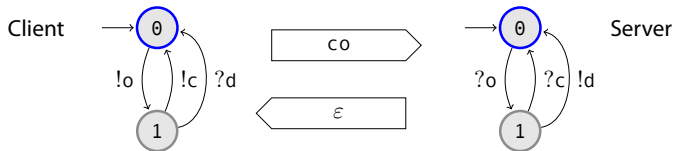
Example: Connection / Disconnection Protocol



$$\langle (\theta, \theta), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o} \langle (1, \theta), (o, \varepsilon) \rangle \xrightarrow{!c}$$

FIFO Systems

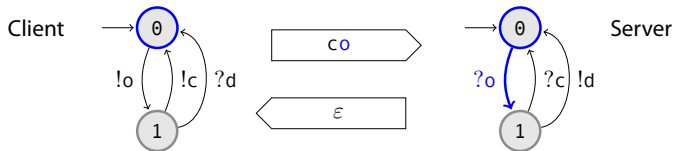
Example: Connection / Disconnection Protocol



$$\langle (\theta, \theta), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o} \langle (1, \theta), (o, \varepsilon) \rangle \xrightarrow{!c} \langle (\theta, \theta), (oc, \varepsilon) \rangle$$

FIFO Systems

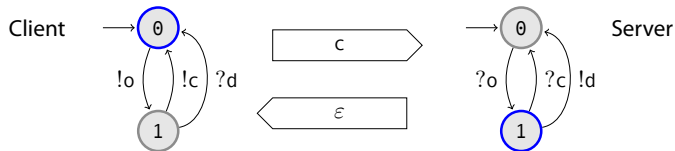
Example: Connection / Disconnection Protocol



$$\langle (\theta, \theta), (\epsilon, \epsilon) \rangle \xrightarrow{!o !c} \langle (\theta, \theta), (oc, \epsilon) \rangle \xrightarrow{?o}$$

FIFO Systems

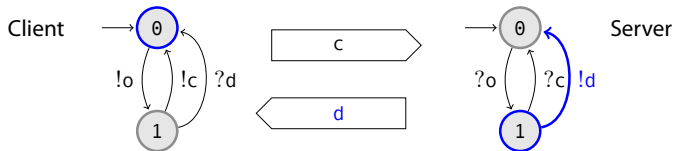
Example: Connection / Disconnection Protocol



$$\langle (\theta, \theta), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o!c} \langle (\theta, \theta), (oc, \varepsilon) \rangle \xrightarrow{?o} \langle (\theta, 1), (c, \varepsilon) \rangle$$

FIFO Systems

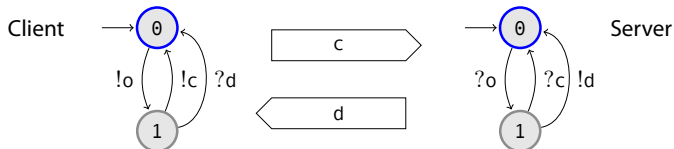
Example: Connection / Disconnection Protocol



$$\langle (\theta, \theta), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o !c} \langle (\theta, \theta), (oc, \varepsilon) \rangle \xrightarrow{?o} \langle (\theta, 1), (c, \varepsilon) \rangle \xrightarrow{!d}$$

FIFO Systems

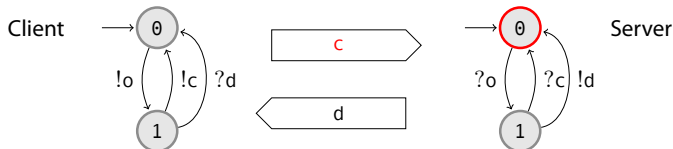
Example: Connection / Disconnection Protocol



$$\langle (\theta, \theta), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o !c} \langle (\theta, \theta), (oc, \varepsilon) \rangle \xrightarrow{?o} \langle (\theta, 1), (c, \varepsilon) \rangle \xrightarrow{!d} \langle (\theta, \theta), (c, d) \rangle$$

FIFO Systems

Example: Connection / Disconnection Protocol

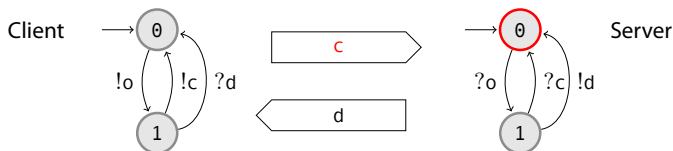


$$\langle (\emptyset, \emptyset), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o !c} \langle (\emptyset, \emptyset), (oc, \varepsilon) \rangle \xrightarrow{?o} \langle (\emptyset, 1), (c, \varepsilon) \rangle \xrightarrow{!d} \langle (\emptyset, \emptyset), (c, d) \rangle$$

⚡ (local) deadlock ⚡
unspecified reception

FIFO Systems

Example: Connection / Disconnection Protocol

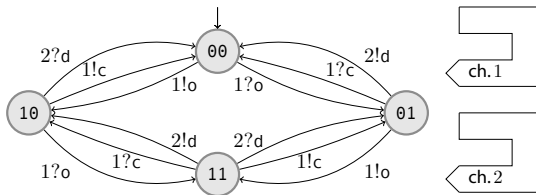


$$\langle (\theta, \theta), (\varepsilon, \varepsilon) \rangle \xrightarrow{!o !c} \langle (\theta, \theta), (oc, \varepsilon) \rangle \xrightarrow{?o} \langle (\theta, 1), (c, \varepsilon) \rangle \xrightarrow{!d} \langle (\theta, \theta), (c, d) \rangle$$

- let $Init = \{(\theta, \theta), (\varepsilon, \varepsilon)\}$
- let $Bad = Q_1 \times \{\theta\} \times c \cdot M^* \times M^*$
- verify **safety**: is Bad not reachable from $Init$?

FIFO Systems

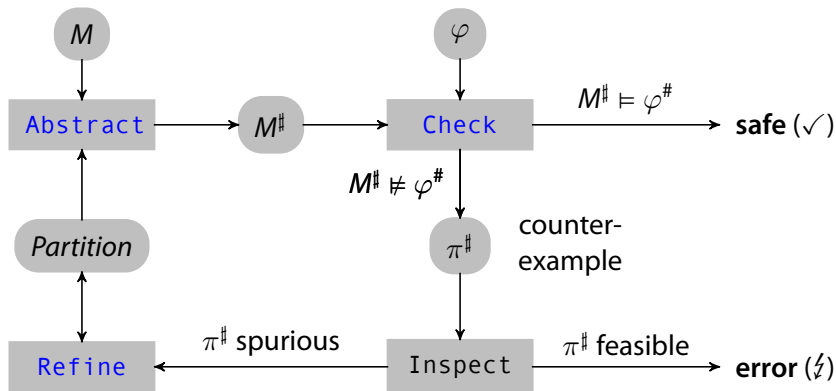
Product Automaton



- asynchronous product automaton
- note: channels need not be point-to-point
- used as our point of departure for CEGAR

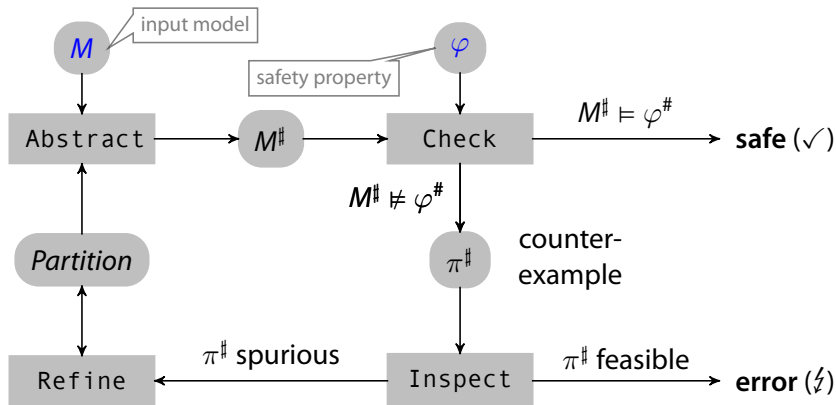
CEGAR

the general approach



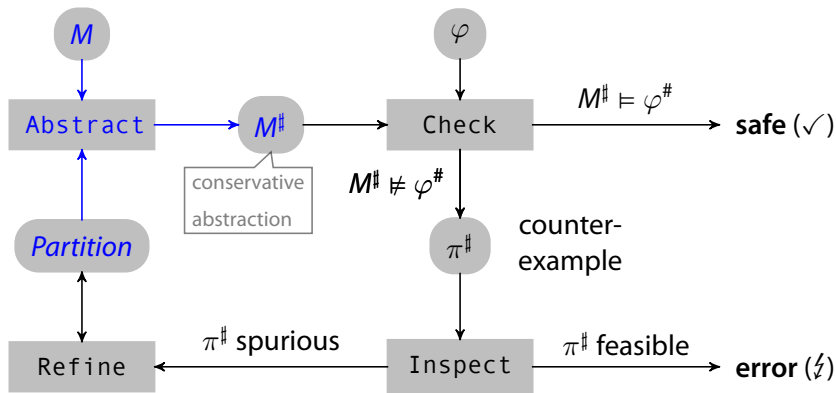
CEGAR

the general approach



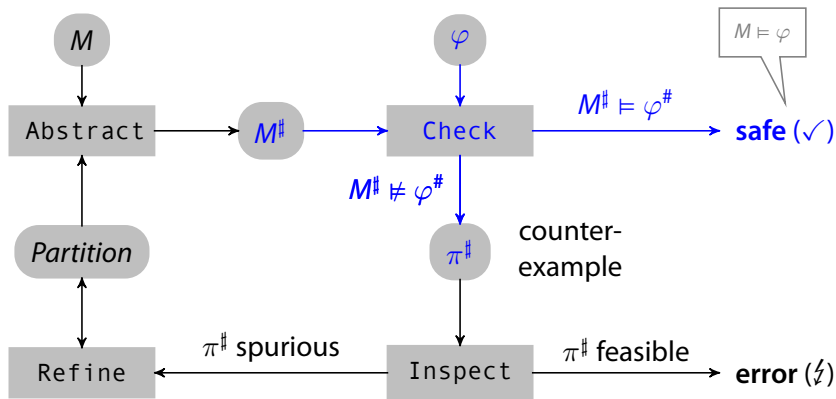
CEGAR

the general approach



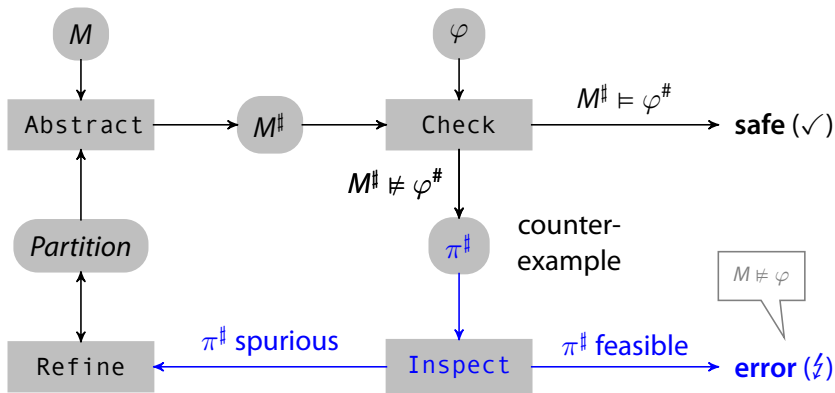
CEGAR

the general approach

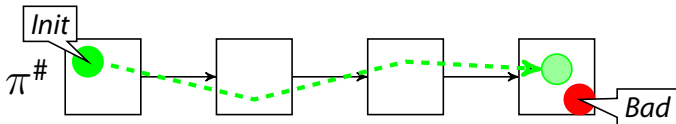


CEGAR

the general approach

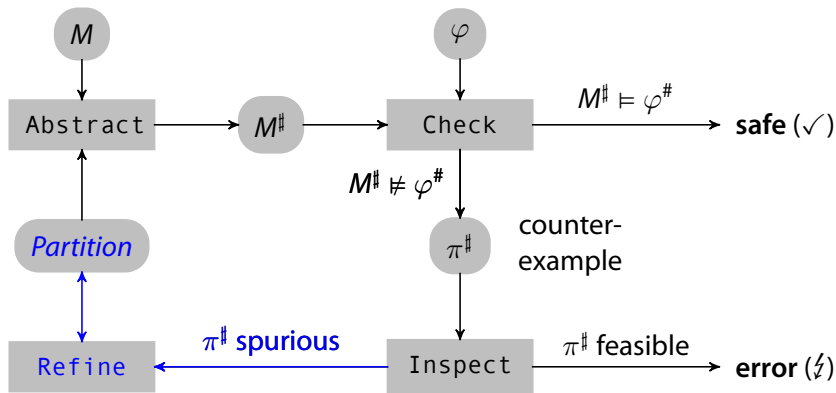


$\varphi^\# \equiv$ set of $Bad^\#$ states not reachable from $Init^\#$



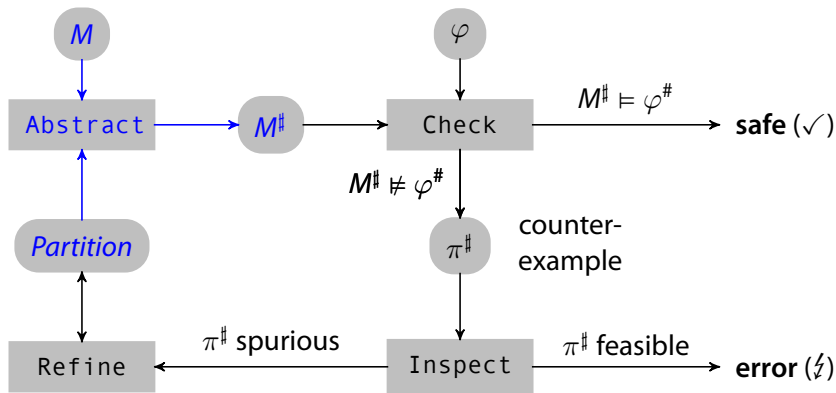
CEGAR

the general approach



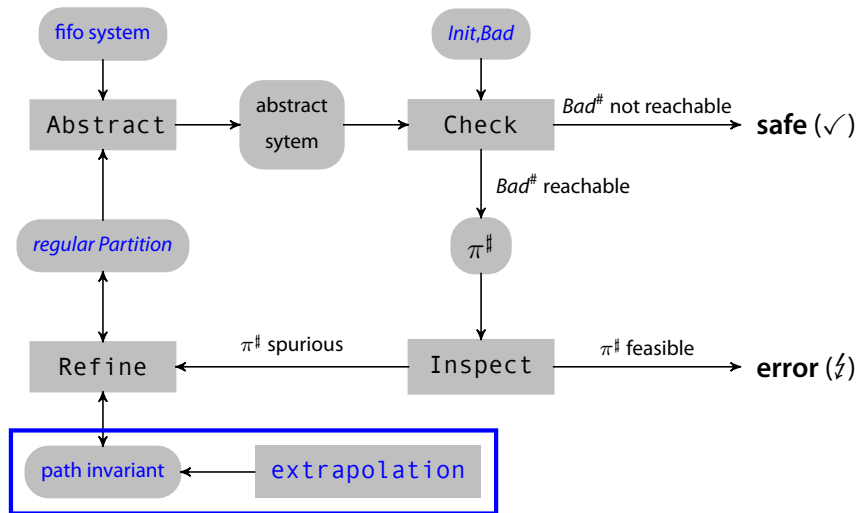
CEGAR

the general approach

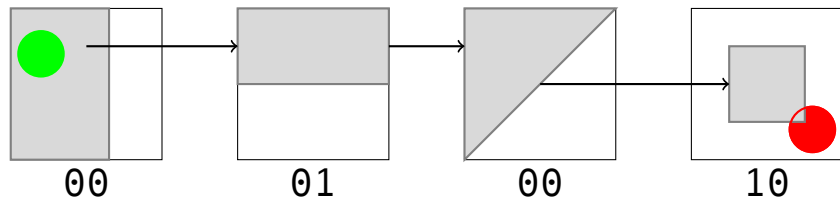


CEGAR

...adapted to our setting



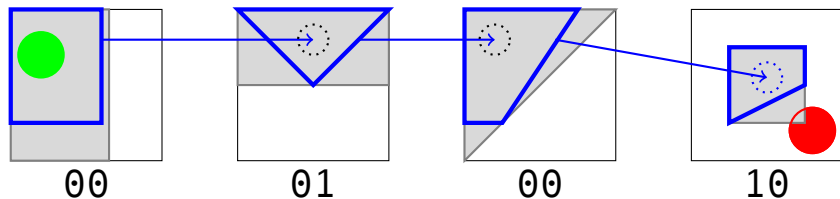
Path Invariants



start with **spurious** abstract counterexample

- initially not disjoint from *Init* ●
- reaches abstract state not disjoint with *Bad* ●

Path Invariants

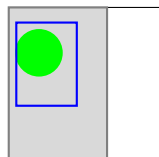


find **path invariant**

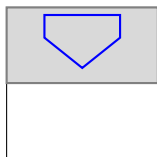
- initially includes *Init* ●
- closed under steps of original counterexample
- finally disjoint with *Bad* ●

recap:
logical invariant for
 $A \not\vdash B$ is sequence
 I_1, \dots, I_n with
(i) $\vdash A \wedge I_1$
(ii) $I_j \vdash I_{j+1}$
(iii) $\not\vdash I_n \wedge B$

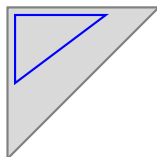
Path Invariants



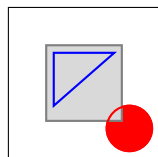
00



01



00

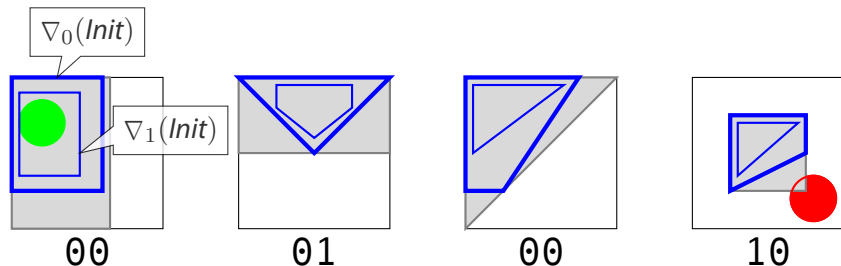


10

given a spurious counterexample, there is a **family** of path invariants

- there must be at least **one**
as the counterexample was shown to be spurious

Path Invariants



try to get **most simple** path invariant **possible**

- apply in each step parametrized **extrapolation** ∇

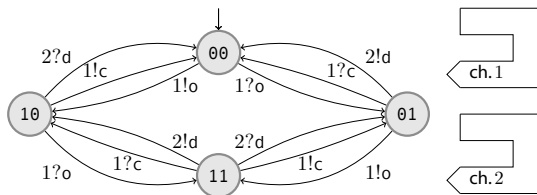
- $\nabla: \mathbb{N} \rightarrow (Rec((M^*)^n) \rightarrow Rec((M^*)^n))$
- $\forall L: \nabla_k(L) \supseteq L$
- $\forall L: \exists k_{max}: \nabla_{k_{max}}(L) = L$

A Practical Example

Safety Verification of the C/D-Protocol

Applying our Algorithm to the Connection / Disconnection Protocol

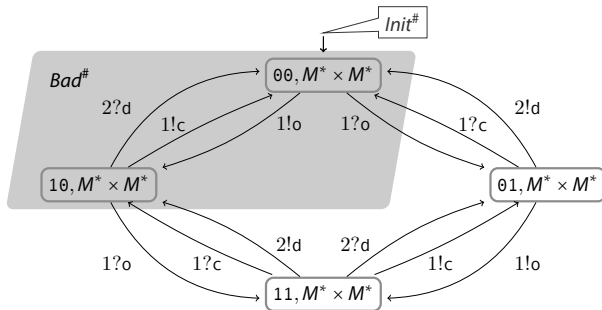
step 0 : basic abstraction



use **partition** abstraction

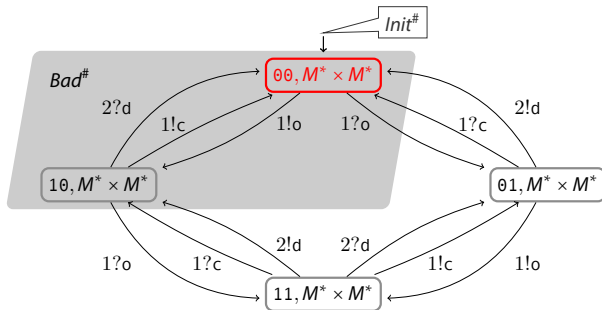
- abstract configurations $\langle\langle (q_1, q_2), \cup_{\text{finite}} L_1 \times L_2 \rangle\rangle$
- L_1, L_2 are **regular** languages over the message alphabet M
- abstract transitions via existential lift...

step 1 : counterexample



- initial partition: $M^* \times M^*$
- calculate $Init^\#$ and $Bad^\#$
- is set of **abstract** configs $Bad^\#$ reachable from $Init^\#$?

step 1 : counterexample



- initial partition: $M^* \times M^*$
- calculate $Init^\#$ and $Bad^\#$
- is set of **abstract** configs $Bad^\#$ reachable from $Init^\#$?
- find simple **counterexample**: $\langle\langle 00, M^* \times M^* \rangle\rangle$

- found counterexample:

$$\langle\langle \emptyset\emptyset, M^* \times M^* \rangle\rangle$$

- counterexample is **spurious**:

$$\langle \emptyset\emptyset, (\varepsilon, \varepsilon) \rangle \notin \text{Bad}$$

- path invariant:

$$(\varepsilon \times \varepsilon)$$

- because



- found counterexample:

$$\langle\langle \emptyset\emptyset, M^* \times M^* \rangle\rangle$$

- counterexample is **spurious**:

$$\langle \emptyset\emptyset, (\varepsilon, \varepsilon) \rangle \notin \text{Bad}$$

- path invariant:

$$(\varepsilon \times \varepsilon)$$

or $\{o^+c \cup \varepsilon\} \times \varepsilon$,
or $o^* \times d^*$,
or ...

- because



- found counterexample:

$$\langle\langle \theta\theta, M^* \times M^* \rangle\rangle$$

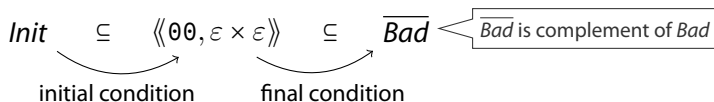
- counterexample is **spurious**:

$$\langle \theta\theta, (\varepsilon, \varepsilon) \rangle \notin \mathit{Bad}$$

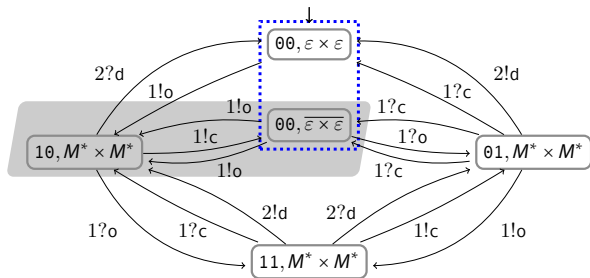
- path invariant:

$$(\varepsilon \times \varepsilon)$$

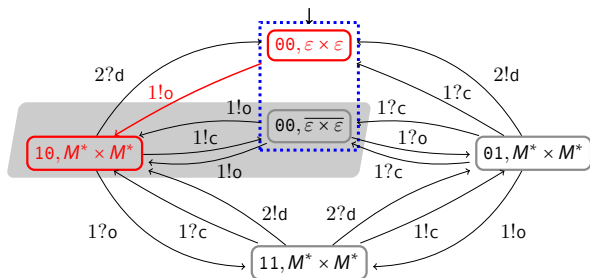
- because



step 2 : counterexample

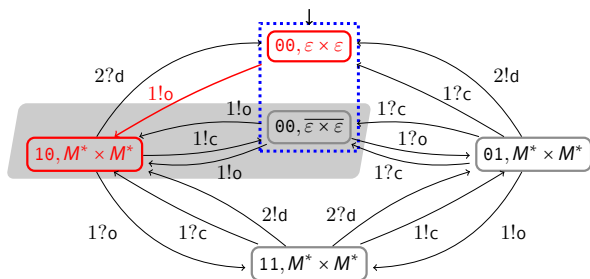


step 2 : counterexample



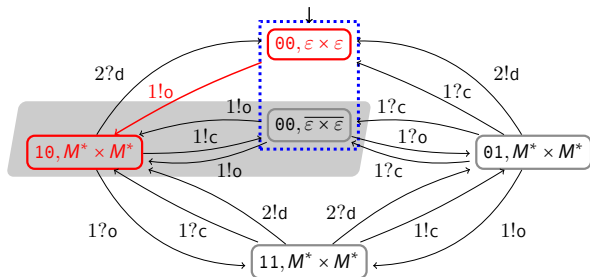
- find counterexample: $\langle\langle 00, \varepsilon \times \varepsilon \rangle\rangle \xrightarrow{!o} \langle\langle 10, M^* \times M^* \rangle\rangle$

step 2 : counterexample



- find counterexample: $\langle\langle 00, \varepsilon \times \varepsilon \rangle\rangle \xrightarrow{!o} \langle\langle 10, M^* \times M^* \rangle\rangle$
- ...is spurious as $\langle 00, (\varepsilon, \varepsilon) \rangle \xrightarrow{!o} \langle 10, (o, \varepsilon) \rangle \notin \text{Bad}$

step 2 : counterexample



- find counterexample: $\langle\langle 00, \varepsilon \times \varepsilon \rangle\rangle \xrightarrow{!o} \langle\langle 10, M^* \times M^* \rangle\rangle$
- ...is spurious as $\langle 00, (\varepsilon, \varepsilon) \rangle \xrightarrow{!o} \langle 10, (o, \varepsilon) \rangle \notin \text{Bad}$
- (simple) path invariant $\varepsilon \times \varepsilon, o \times \varepsilon$

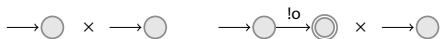
- apply **extrapolation**

$$\varepsilon \times \varepsilon$$

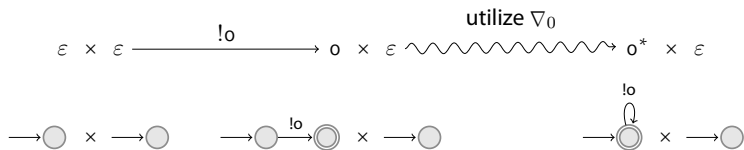


- apply **extrapolation**

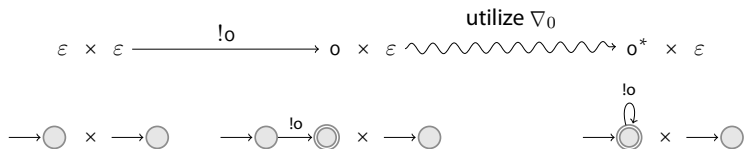
$$\varepsilon \times \varepsilon \xrightarrow{!o} 0 \times \varepsilon$$



- apply **extrapolation**



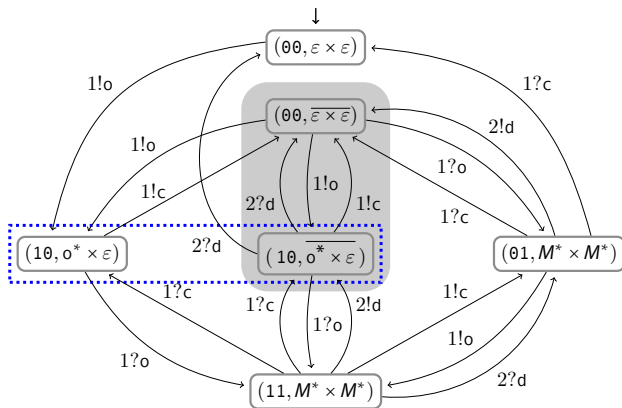
- apply **extrapolation**



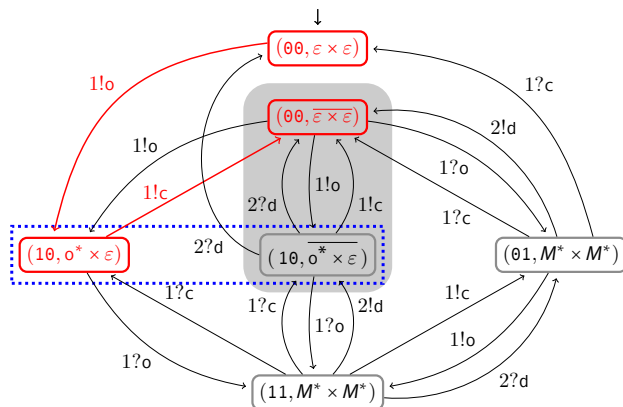
- extrapolated path invariant $\varepsilon \times \varepsilon, o^* \times \varepsilon$



step 3 : counterexample



step 3 : counterexample



- (spurious) counterexample:

$$\langle\langle 00, \varepsilon \times \varepsilon \rangle\rangle \xrightarrow{!o} \langle\langle 10, o^* \times \varepsilon \rangle\rangle \xrightarrow{!c} \langle\langle 00, \bar{\varepsilon} \times \bar{\varepsilon} \rangle\rangle$$

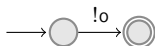
- path invariant via extrapolation: utilize ∇_0

$$\varepsilon(x, \varepsilon)$$



- path invariant via extrapolation: utilize ∇_0

$$\varepsilon(x \ \varepsilon) \xrightarrow{!_0}$$



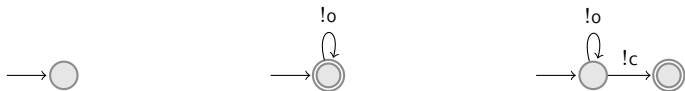
- path invariant via extrapolation: utilize ∇_0

$$\varepsilon(x \ \varepsilon) \xrightarrow{!o} o^*(x \ \varepsilon)$$



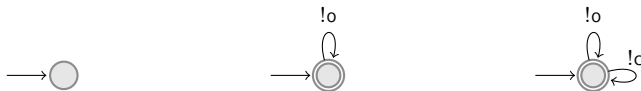
- path invariant via extrapolation: utilize ∇_0

$$\varepsilon(x \ \varepsilon) \xrightarrow{!o} o^*(x \ \varepsilon) \xrightarrow{!c} \rightarrow$$



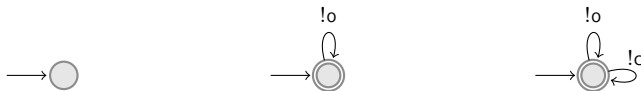
- path invariant via extrapolation: utilize ∇_0

$$\varepsilon (x \ \varepsilon) \xrightarrow{!o} o^* (x \ \varepsilon) \xrightarrow{!c} \{o, c\}^* (x \ \varepsilon)$$



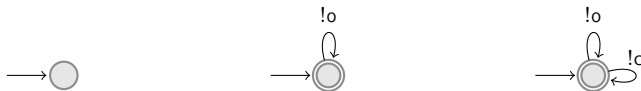
- path invariant via extrapolation: utilize ∇_0

$$\varepsilon (x \ \varepsilon) \xrightarrow{!o} o^* (x \ \varepsilon) \xrightarrow{!c} \{o, c\}^* (x \ \varepsilon) \quad \text{⚡ Bad}$$



- path invariant via extrapolation: utilize ∇_0

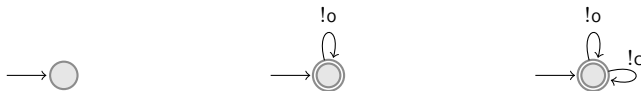
$$\varepsilon (x \ \varepsilon) \xrightarrow{!o} o^* (x \ \varepsilon) \xrightarrow{!c} \{o, c\}^* (x \ \varepsilon) \quad \text{⚡ Bad}$$



- use **finer** extrapolation: ∇_1

- path invariant via extrapolation: utilize ∇_0

$$\varepsilon (x \ \varepsilon) \xrightarrow{!o} o^* (x \ \varepsilon) \xrightarrow{!c} \{o, c\}^* (x \ \varepsilon) \quad \text{⚡ Bad}$$



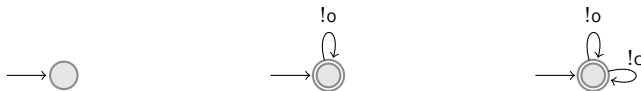
- use **finer** extrapolation: ∇_1

$$\varepsilon (x \ \varepsilon)$$



- path invariant via extrapolation: utilize ∇_0

$$\varepsilon (\times \varepsilon) \xrightarrow{!o} o^* (\times \varepsilon) \xrightarrow{!c} \{o, c\}^* (\times \varepsilon) \quad \text{⚡ Bad}$$



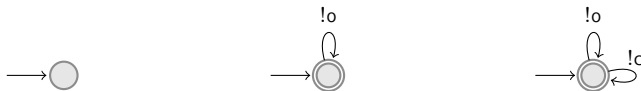
- use **finer** extrapolation: ∇_1

$$\varepsilon (\times \varepsilon) \xrightarrow{!o} o (\times \varepsilon)$$



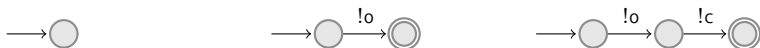
- path invariant via extrapolation: utilize ∇_0

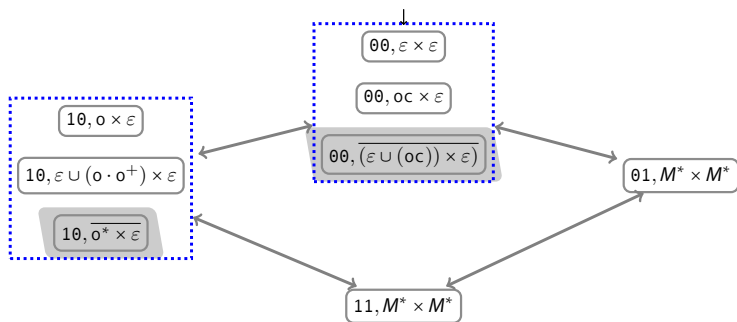
$$\varepsilon (\times \varepsilon) \xrightarrow{!o} o^* (\times \varepsilon) \xrightarrow{!c} \{o, c\}^* (\times \varepsilon) \quad \text{⚡ Bad}$$



- use **finer** extrapolation: ∇_1

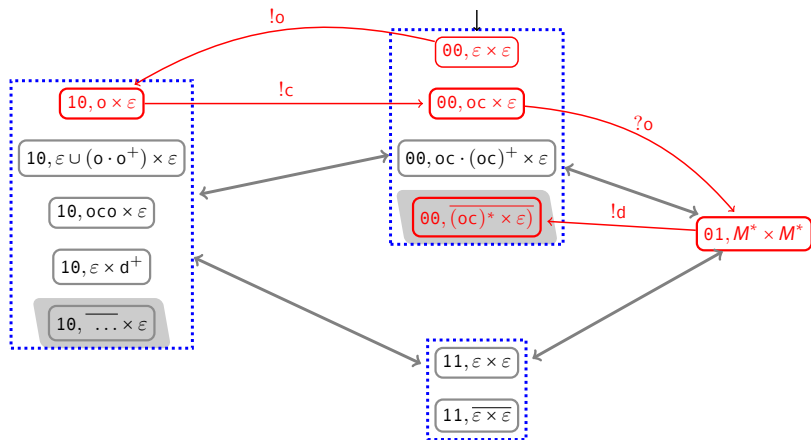
$$\varepsilon (\times \varepsilon) \xrightarrow{!o} o (\times \varepsilon) \xrightarrow{!c} oc (\times \varepsilon) \quad \checkmark \text{ path invariant}$$





- check, inspect, refine...

...step 8 : counterexample



- counterexample is **feasible** ! hence, *Bad* is reachable ↯

Technical Contributions

- parametrized extrapolation

$$\nabla: \mathbb{N} \rightarrow (\text{Rec}((M^*)^n) \rightarrow \text{Rec}((M^*)^n))$$

- use finite automata quotienting
wrt. colored bisimulation of parametrized depth
- different algorithms for path invariant generation
 - single-split, split along the path-invariant
 - forward, backward variants
- (partial) termination results
 - if fifo system is unsafe and applying bfs search for $\pi^\#$
 - if reachability set is finite, ∇ restricted, and bfs search for $\pi^\#$

Technical Contributions

- parametrized extrapolation

$$\nabla: \mathbb{N} \rightarrow (\text{Rec}((M^*)^n) \rightarrow \text{Rec}((M^*)^n))$$

- use finite automata quotienting
wrt. colored bisimulation of parametrized depth
- different algorithms for path invariant generation
 - single-split, split along the path-invariant
 - forward, backward variants
- (partial) termination results
 - if fifo system is unsafe and applying bfs search for $\pi^\#$
 - if reachability set is finite, ∇ restricted, and bfs search for $\pi^\#$

Technical Contributions

- parametrized extrapolation

$$\nabla: \mathbb{N} \rightarrow (\text{Rec}((M^*)^n) \rightarrow \text{Rec}((M^*)^n))$$

- use finite automata quotienting
wrt. colored bisimulation of parametrized depth
- different algorithms for path invariant generation
 - single-split, split along the path-invariant
 - forward, backward variants
- (partial) termination results
 - if fifo system is unsafe and applying bfs search for $\pi^\#$
 - if reachability set is finite, ∇ restricted, and bfs search for $\pi^\#$

Comparison to (some) Other Approaches

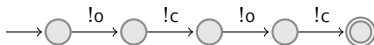
- acceleration based approaches
 - Lash/QDD, TReX/SRE
 - no counterexamples
 - our approach mimics acceleration

acceleration !!!! but we get same results...

no

Comparison to (some) Other Approaches

- acceleration based approaches
 - Lash/QDD, TReX/SRE
 - no counterexamples
 - our approach mimics acceleration

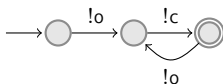


acceleration !!!! but we get same results...

no

Comparison to (some) Other Approaches

- acceleration based approaches
 - Lash/QDD, TReX/SRE
 - no counterexamples
 - our approach mimics acceleration



application of ∇_1
(bisimulation of depth 1)
leads to $(oc)^+$

no

acceleration !!!! but we get same results...

Comparison to (some) Other Approaches

- acceleration based approaches
 - Lash/QDD, TReX/SRE
 - no counterexamples
 - our approach mimics acceleration
- ARMC - Abstract Regular Model Checking
 - approximative fixpoint iteration based on regular languages
 - global refinement / uniform precision
- our approach
 - local refinement / adaptive precision
 - flexible wrt. extrapolation and invariant generation
 - generic (not bound to automata with "regular" data)
 - counterexample guided

Comparison to (some) Other Approaches

- acceleration based approaches
 - Lash/QDD, TReX/SRE
 - no counterexamples
 - our approach mimics acceleration
- ARMC - Abstract Regular Model Checking
 - approximative fixpoint iteration based on regular languages
 - global refinement / uniform precision
- our approach
 - local refinement / adaptive precision
 - flexible wrt. extrapolation and invariant generation
 - generic (not bound to automata with "regular" data)
 - counterexample guided

Comparison to (some) Other Approaches

- acceleration based approaches
 - Lash/QDD, TReX/SRE
 - no counterexamples
 - our approach mimics acceleration
- ARMC - Abstract Regular Model Checking
 - approximative fixpoint iteration based on regular languages
 - global refinement / uniform precision
- our approach
 - local refinement / adaptive precision
 - flexible wrt. extrapolation and invariant generation
 - generic (not bound to automata with "regular" data)
 - counterexample guided

Empirical Evaluation

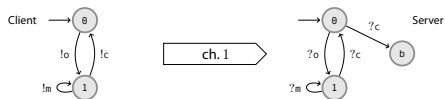
benchmarks...

control automaton!

protocol	states/trans.	time [s]	mem [MiB]	loops	states [#] /trans [#]
ABP	16/64	2.13	1.58	208	274/1443
c/d protocol	5/17	0.01	0.61	6	11/32
nested c/d protocol	6/17	1.15	1.09	93	100/339
non-regular protocol	9/18	0.06	0.61	14	25/39
Peterson	10648/56628	2.14	32.09	51	10709/56939
(simplified) Tcp	196/588	1.38	2.06	183	431/1439
server with 2 clients	255/2160	9.61	4.97	442	731/7383
token ring	625/4500	4.57	6.42	319	1004/6956
sliding window	225/2010	0.93	2.55	148	388/2367

benchmarks...

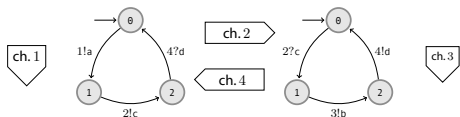
protocol	states/trans.	time [s]	mem [MiB]	loops	states [#] /trans [#]
ABP	16/64	2.13	1.58	208	274/1443
c/d protocol	5/17	0.01	0.61	6	11/32
nested c/d protocol	6/17	1.15	1.09	93	100/339
non-regular protocol	9/18	0.06	0.61	14	25/39
Peterson	10648/56628	2.14	32.09	51	10709/56939
(simplified) Tcp	196/588	1.38	2.06	183	431/1439
server with 2 clients	255/2160	9.61	4.97	442	731/7383
token ring	625/4500	4.57	6.42	319	1004/6956
sliding window	225/2010	0.93	2.55	148	388/2367



⚡ nested loops and acceleration ⚡
(TRex does not terminate in reasonable time)

benchmarks...

protocol	states/trans.	time [s]	mem [MiB]	loops	states [#] /trans [#]
ABP	16/64	2.13	1.58	208	274/1443
c/d protocol	5/17	0.01	0.61	6	11/32
nested c/d protocol	6/17	1.15	1.09	93	100/339
non-regular protocol	9/18	0.06	0.61	14	25/39
Peterson	10648/56628	2.14	32.09	51	10709/56939
(simplified) Tcp	196/588	1.38	2.06	183	431/1439
server with 2 clients	255/2160	9.61	4.97	442	731/7383
token ring	625/4500	4.57	6.42	319	1004/6956
sliding window	225/2010	0.93	2.55	148	388/2367



use ch. 1 and ch. 3 as "counters"

(our implementation of ARMC does not terminate in reasonable time)

Summary

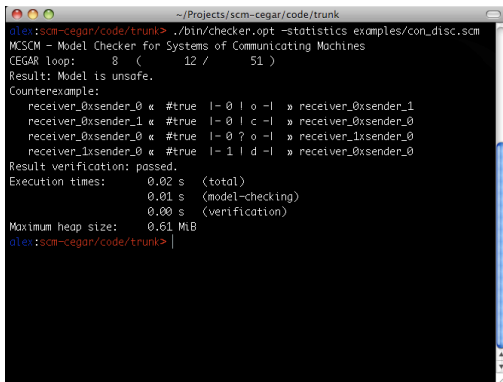
- CEGAR for safety verification of infinite state systems
 - generic method
 - based on path invariants
 - and extrapolation
- adapted to fifo systems
 - by encoding partitions as regular languages
 - and utilizing parametrized colored bisimulation equivalence based quotienting for extrapolation
- implemented in tool McScM

Perspective

- adapt to other classes of infinite state systems
(WSTS, hybrid automata, parametrized systems,...)
- additional termination results possible ?
(for decidable classes, e.g., lossy fifo systems)
- exchange data over infinite domain over channels
- "real-life" examples (e.g. Socket API based applications)
- ... (feel free to append your ideas)

M_cS_cM

- get it at <https://www.labri.fr/~heussner/mcscm>
- based on libraries from Tristan Le Gall & Bertrand Jeannet (thanks!)
- BSD-style licence
- programmed in OCaml
- includes all examples of this talk
- binary release for the impatient
- coffee-pause demo on demand!



```
~/Projects/scm-cegar/code/trunk
alex@scm-cegar/code/trunk> ./bin/checker.opt -statistics examples/con_disc.scm
MCSCM - Model Checker for Systems of Communicating Machines
CEGAR loop: 8 ( 12 / 51 )
Result: Model is unsafe.
Counterexample:
  receiver_0xsender_0 « #true | - 0 | o - | » receiver_0xsender_1
  receiver_0xsender_1 « #true | - 0 | c - | » receiver_0xsender_0
  receiver_0xsender_0 « #true | - 0 ? o - | » receiver_1xsender_0
  receiver_1xsender_0 « #true | - 1 | d - | » receiver_0xsender_0
Result verification: passed.
Execution times:      0.02 s (total)
                    0.01 s (model-checking)
                    0.00 s (verification)
Maximum heap size:   0.61 MiB
alex@scm-cegar/code/trunk> |
```

M_cS_cM

- get it at <https://www.labri.fr/~heussner/mcscm>
- based on libraries from Tristan Le Gall & Bertrand Jeannet (thanks!)
- BSD-style licence
- programmed in OCaml
- includes all examples of this talk
- binary release for the impatient
- coffee-pause demo on demand!

```
~/Projects/scm-cegar/code/trunk
alex@scm-cegar/code/trunk> ./bin/checker.opt -statistics examples/con_disc.scm
MCSCM - Model Checker for Systems of Communicating Machines
CEGAR loop: 8 ( 12 / 51 )
Result: Model is unsafe.
Counterexample:
  receiver_0xsender_0 « #true | - 0 | o - | » receiver_0xsender_1
  receiver_0xsender_1 « #true | - 0 | c - | » receiver_0xsender_0
  receiver_0xsender_0 « #true | - 0 ? o - | » receiver_1xsender_0
  receiver_1xsender_0 « #true | - 1 | d - | » receiver_0xsender_0
Result verification: passed.
Execution times:      0.02 s (total)
                    0.01 s (model-checking)
                    0.00 s (verification)
Maximum heap size:   0.61 MiB
alex@scm-cegar/code/trunk> |
```