INVARIANTS FOR FINITE INSTANCES AND BEYOND

October, 21st 2013

Sylvain Conchon, Amit Goel, Sava Kristić,
Alain Mebsout, Fatiha Zaïdi

LRI, Université Paris-Sud
Strategic CAD Labs, Intel Corporation

Challenge

How to prove safety of industrial size protocols like FLASH for an arbitrary number of processes ?

Challenge

How to prove safety of industrial size protocols like FLASH for an arbitrary number of processes ?

- automatically

The FLASH protocol

Stanford FLASH multiprocessor architecture (1994)

- Cache-coherence shared memory
- High-performance message passing
- Industrial size: 67 million states for 4 processes (28,000 states for German)

The FLASH protocol

Stanford FLASH multiprocessor architecture (1994)

- Cache-coherence shared memory
- High-performance message passing
- Industrial size: 67 million states for 4 processes (28,000 states for German)

Who proved the protocol?

- Park and Dill, 1996, PVS proof
- Das, Dill and Park, 1999, by predicate abstraction
- McMillan, 2001, by compositional model checking
- Chou, Mannava, Park, 2004, CMP method inspired by McMillan's work
- Talapur and Tuttle, 2008, message-flows extension of CMP

None of these proofs are purely automatic

Solutions

- Model checking of parameterized systems
- Decidable fragment
- Cubicle implements backward reachability

Solutions

- Model checking of parameterized systems
- Decidable fragment
- Cubicle implements backward reachability

Does it work ?

Some benchmarks

	Cubicle	CMurphi		
Szymanski_at	0.30s	8.04s (8)	5m12s (10)	2h50m (12)
German_Baukus	7.03s	0.74s (4)	19m35s (8)	4h49m (10)
German.CTC	3m23s	1.83s (4)	43m46s (8)	12h35m (10)
German_pfs	3m58s	0.99s (4)	22m56s (8)	5h30m (10)
Chandra-Toueg	2h01m	5.68s (4)	2m58s (5)	1h36m (6)

Some benchmarks

	Cubicle	CMurphi		
Szymanski_at	0.30s	8.04s (8)	5m12s (10)	2h50m (12)
German_Baukus	7.03s	0.74s (4)	19m35s (8)	4h49m (10)
German.CTC	3m23s	1.83s (4)	43m46s (8)	12h35m (10)
German_pfs	3m58s	0.99s (4)	22m56s (8)	5h30m (10)
Chandra-Toueg	2h01m	5.68s (4)	2m58s (5)	1h36m (6)
Szymanski_na	T.O.	0.88s (4)	8m25s (6)	7h08m (8)
Flash_nodata	O.M.	4.86s (3)	3m33s (4)	2h46m (5)
Flash	O.M.	1m27s (3)	2h15m (4)	O.M. (5)

O.M. $>$ 20 GB

T.O. $>$ 20 h

How to scale ?

- Reduce the state space to explore
- Invariants for parameterized case
- Interesting behaviors often observable on small instances

Invariants inference

Problem: Invariants often harder to prove than original property

Invariants inference

Problem: Invariants often harder to prove than original property

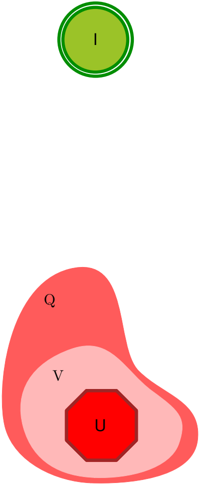
Idea: use finite instances to infer invariants for parametrized case

- Insert and check on the fly in backward reachability loop
- Backtrack if necessary

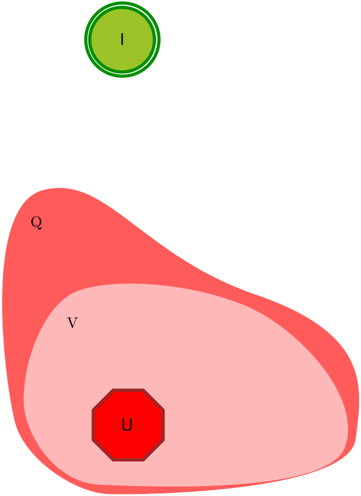
BRAB: **B**ackward **R**eachability with **A**pproximations and **B**acktracking

Backward reachability algorithm

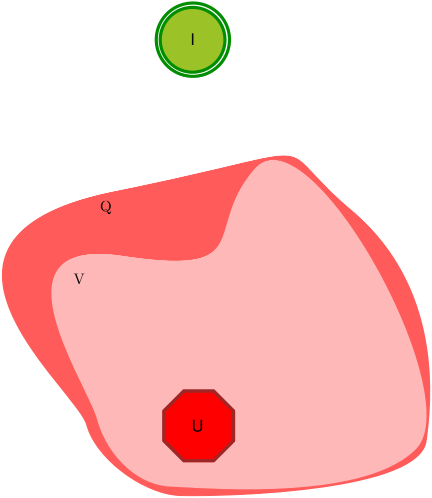
Backward reachability algorithm



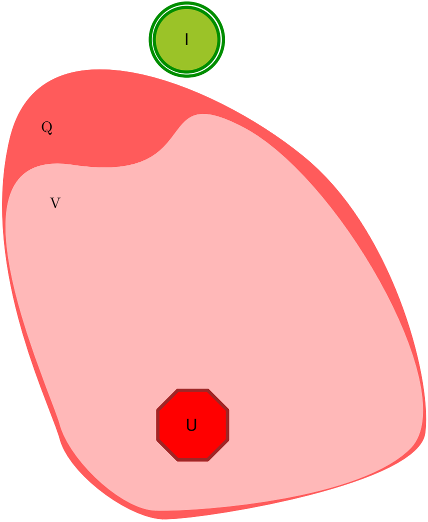
Backward reachability algorithm



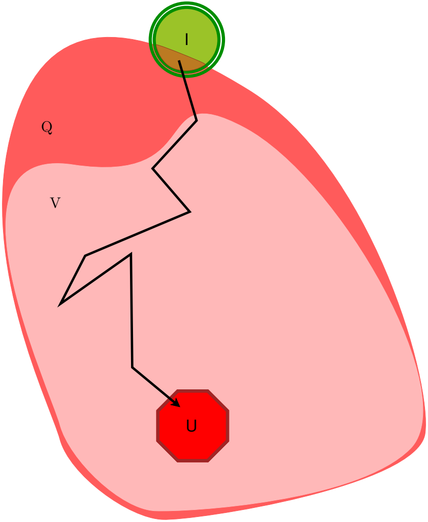
Backward reachability algorithm



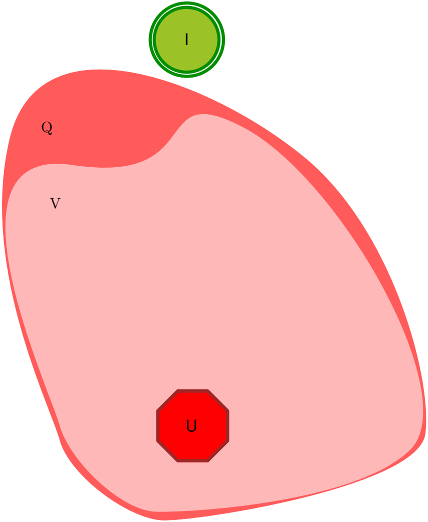
Backward reachability algorithm



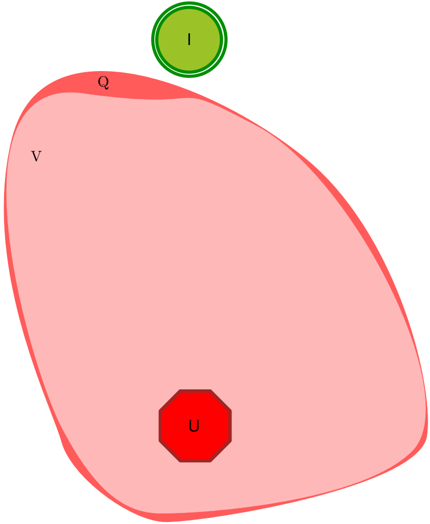
Backward reachability algorithm



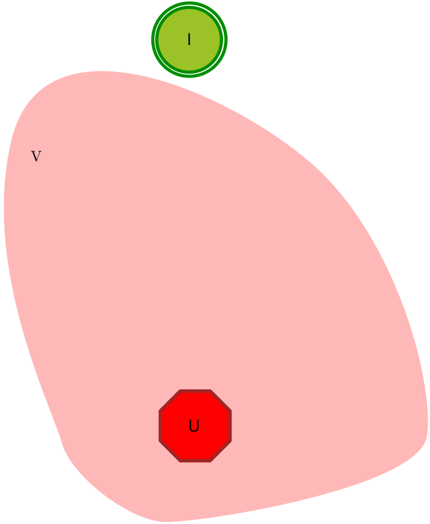
Backward reachability algorithm



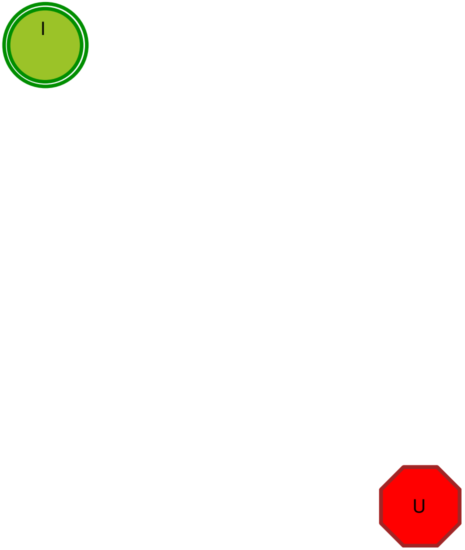
Backward reachability algorithm



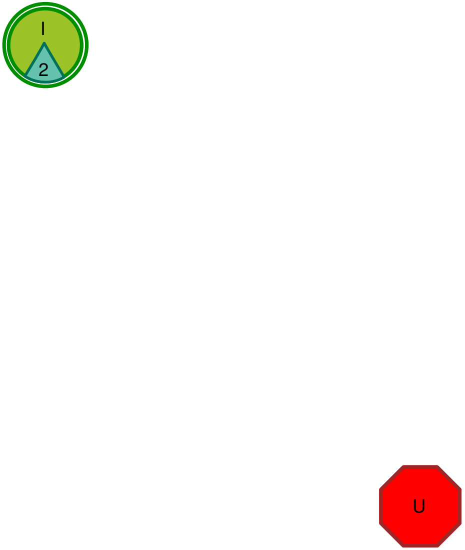
Backward reachability algorithm



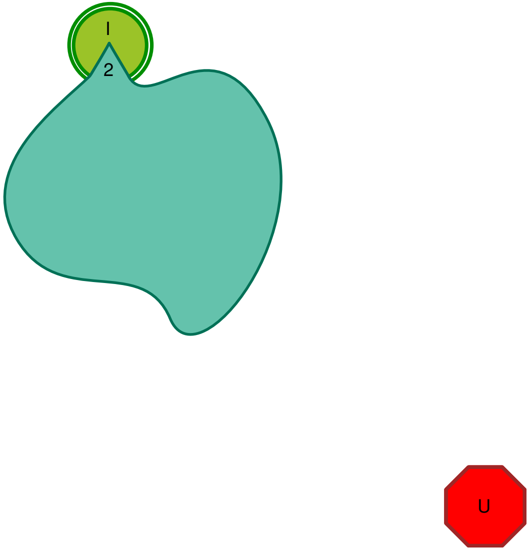
BRAB: intuition



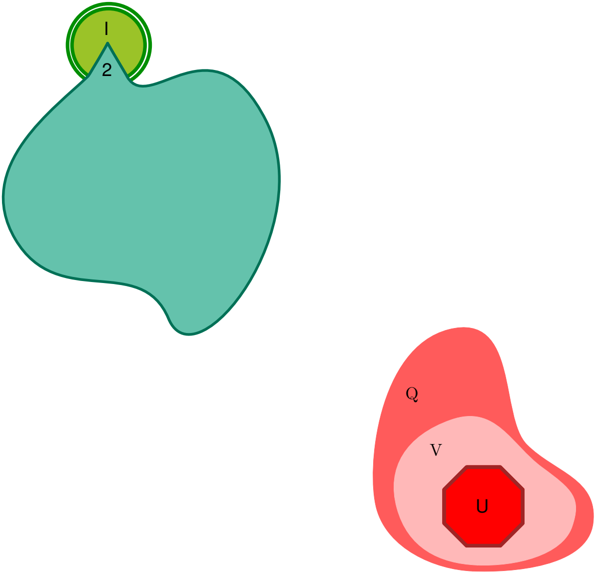
BRAB: intuition



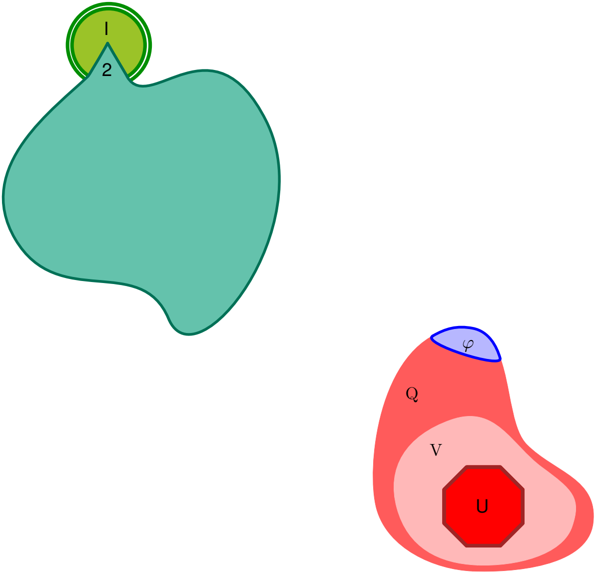
BRAB: intuition



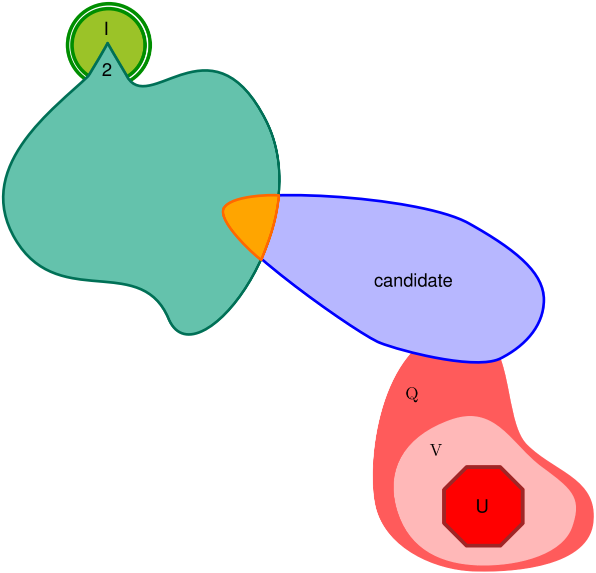
BRAB: intuition



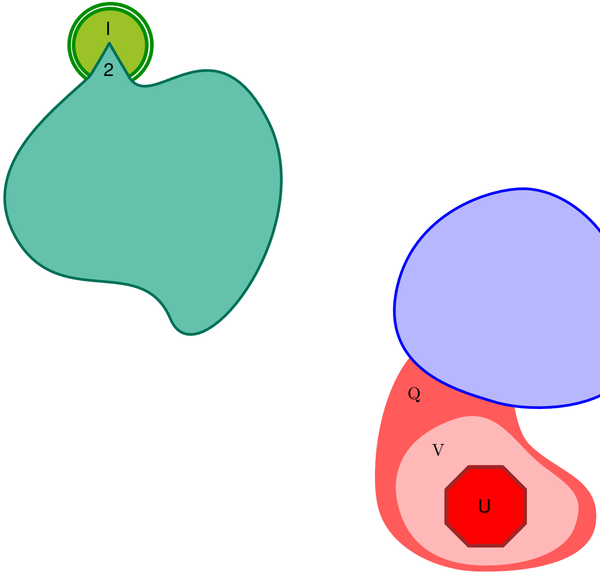
BRAB: intuition



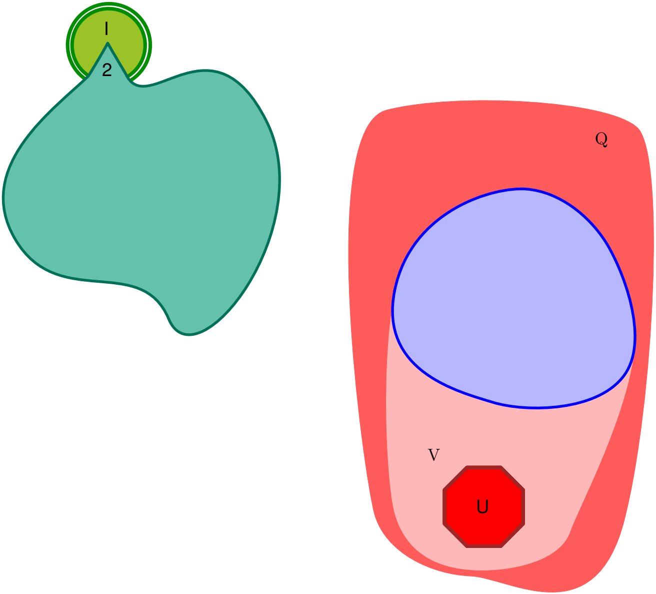
BRAB: intuition



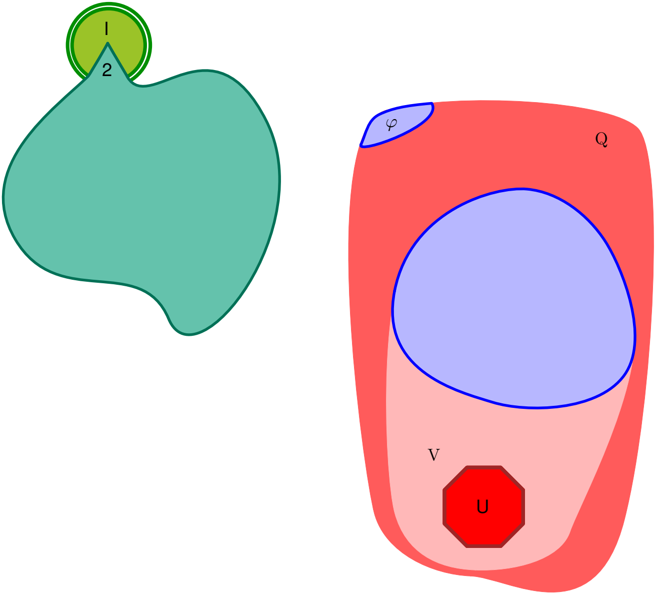
BRAB: intuition



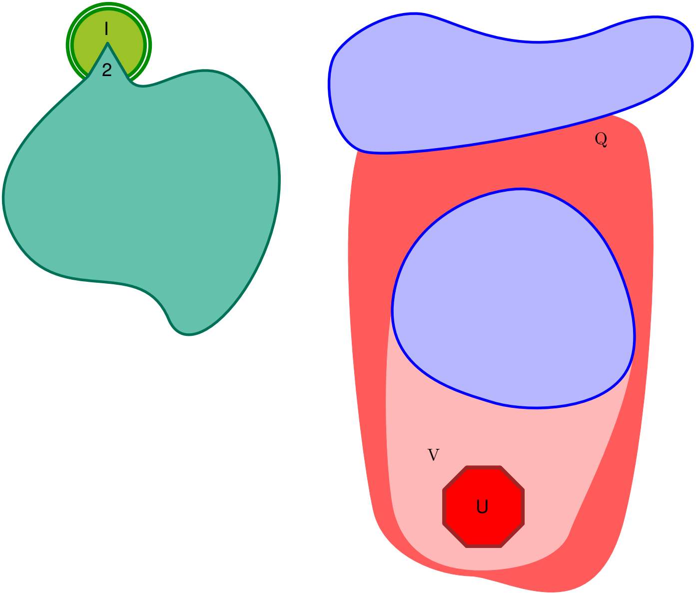
BRAB: intuition



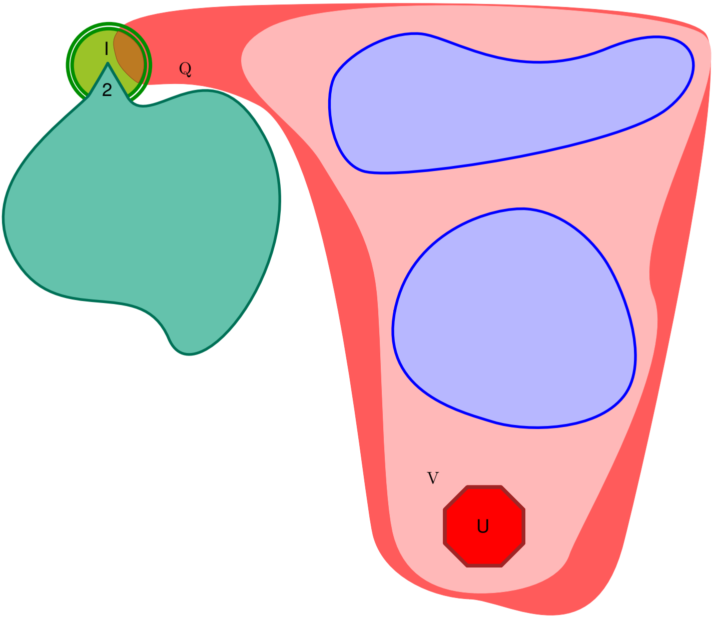
BRAB: intuition



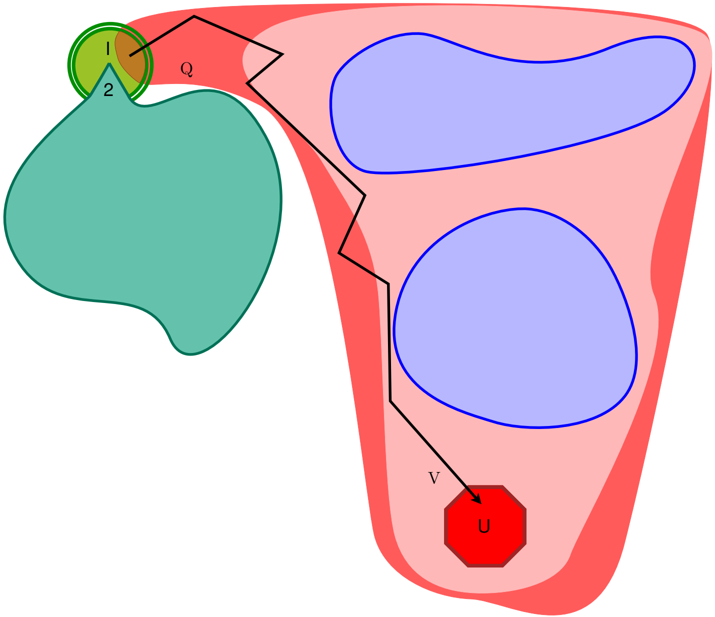
BRAB: intuition



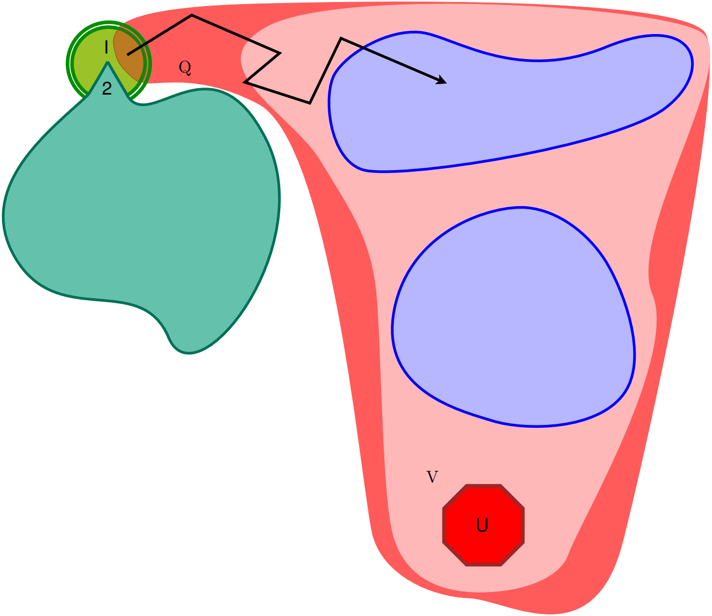
BRAB: intuition



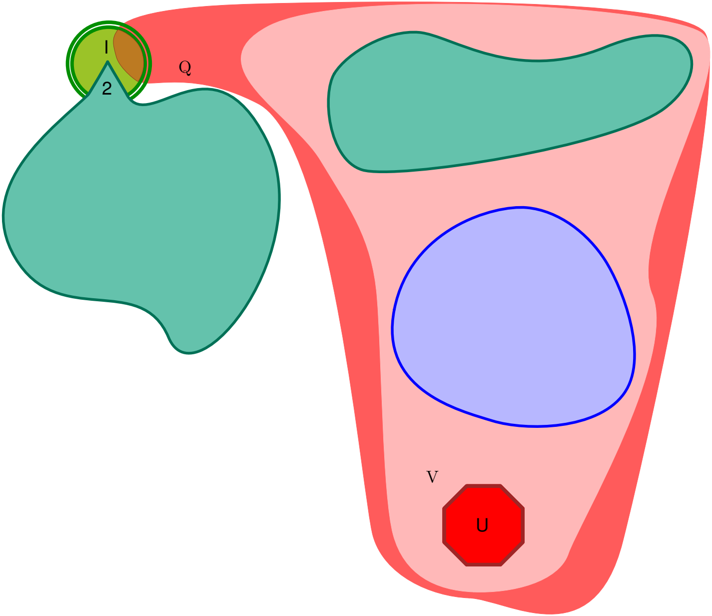
BRAB: intuition



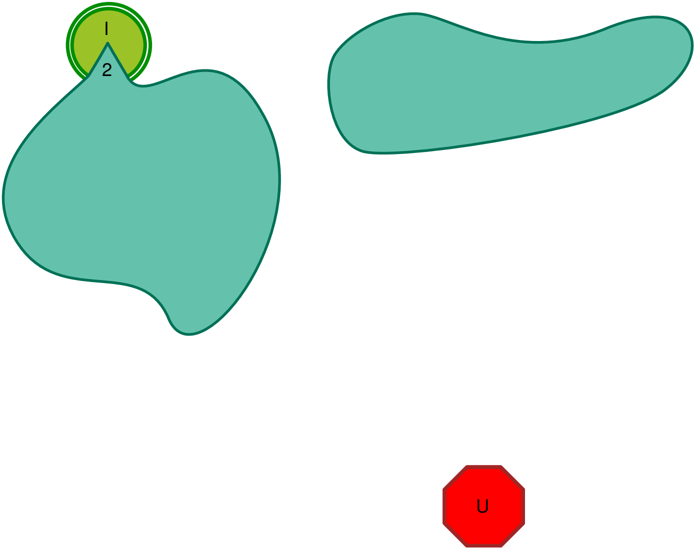
BRAB: intuition



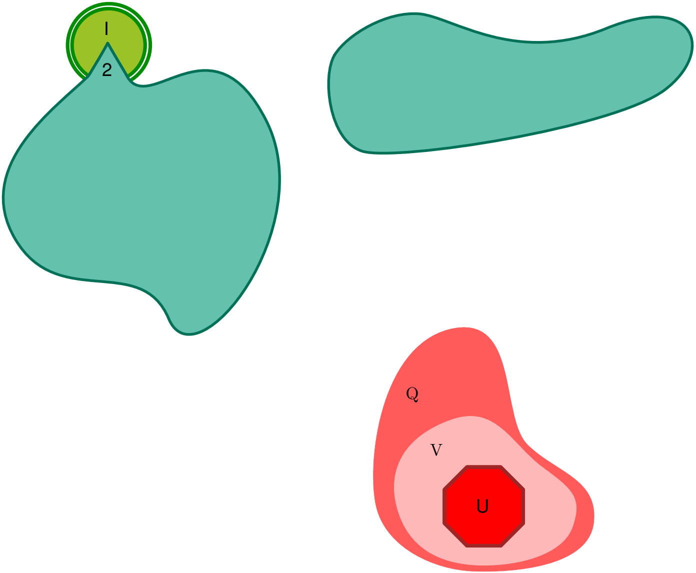
BRAB: intuition



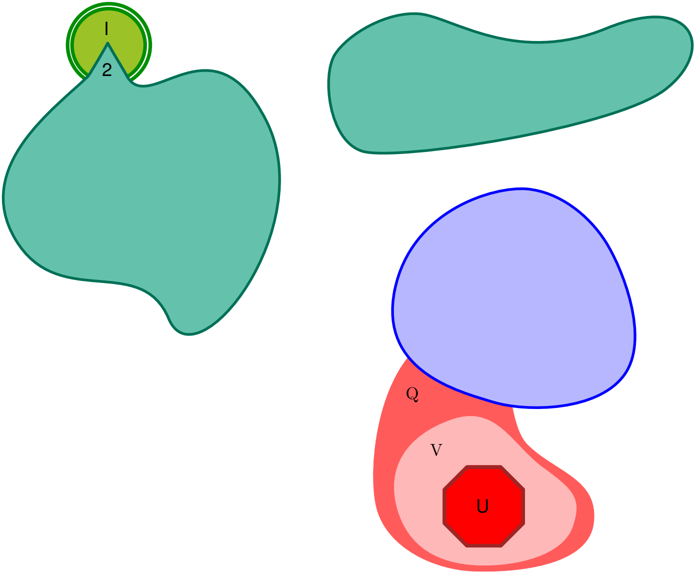
BRAB: intuition



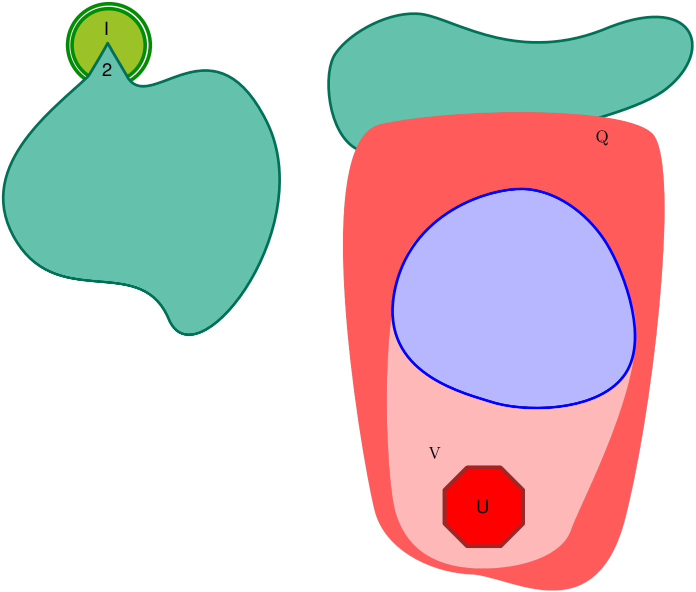
BRAB: intuition



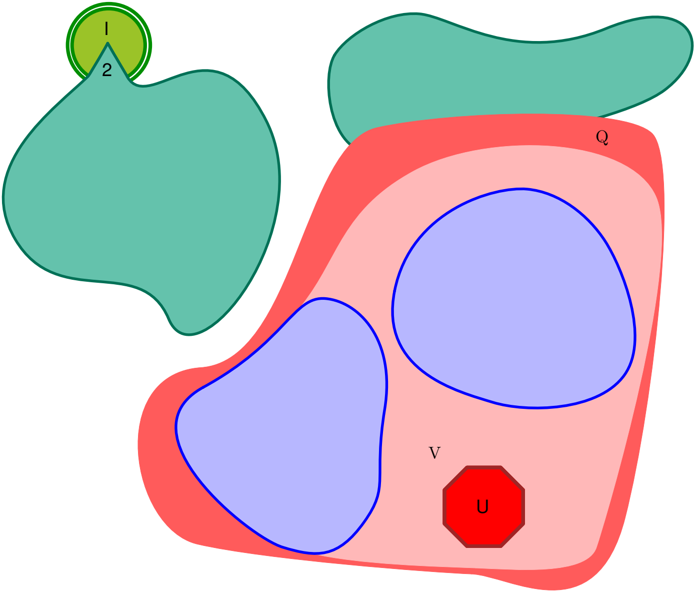
BRAB: intuition



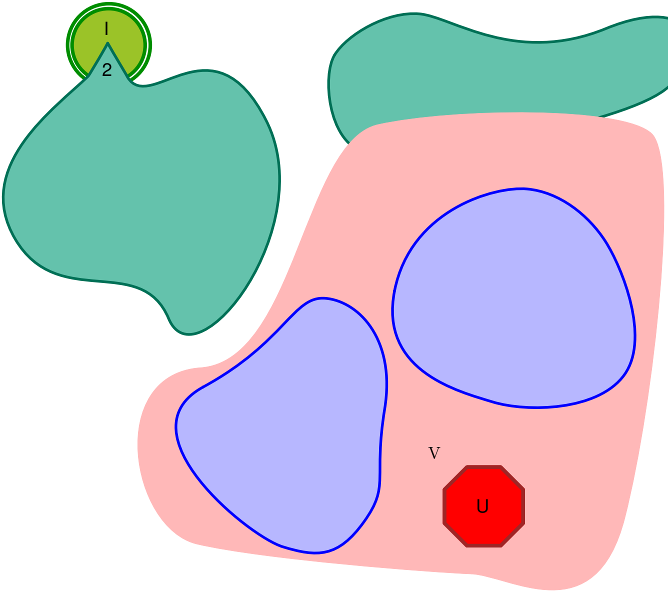
BRAB: intuition



BRAB: intuition



BRAB: intuition



Framework

- Symbolic framework for parameterized systems
- States : formulas in a decidable fragment of FOL
- Pre-image effectively computable
- Post-image effectively computable for a finite instance

Framework

- Symbolic framework for parameterized systems
- States : formulas in a decidable fragment of FOL
- Pre-image effectively computable
- Post-image effectively computable for a finite instance

In Cubicle $\rightarrow$ array-based transition systems

Example: German-*ish* cache coherence protocol

Client i:

$\texttt{Cache}[i] \in \{\mathsf{E}, \mathsf{S}, \mathsf{I}\}$

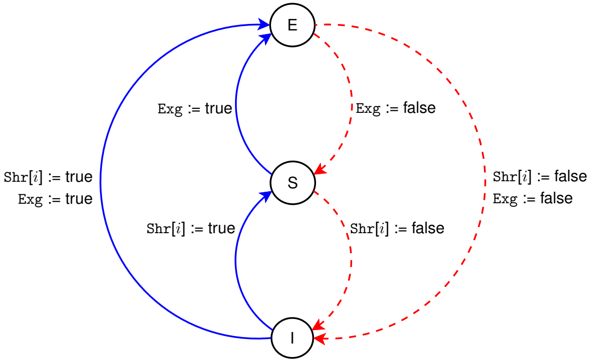
Directory:

$\texttt{Cmd} \in \{\mathsf{rs}, \mathsf{re}, \epsilon\}$

$\texttt{Ptr} \in proc$

$\texttt{Shr}[i] \in \{\mathsf{true}, \mathsf{false}\}$

$\texttt{Exg} \in \{\mathsf{true}, \mathsf{false}\}$



Initial states: $\forall i.\ \texttt{Cache}[i] = \mathsf{I} \ \wedge\ \neg\texttt{Shr}[i] \ \wedge\ \neg\texttt{Exg} \ \wedge\ \texttt{Cmd} = \epsilon$

Unsafe states: $\exists i, j.\ i \neq j \ \wedge\ \texttt{Cache}[i] = \mathsf{E} \ \wedge\ \texttt{Cache}[j] \neq \mathsf{I}$ **?**

(cubes)

Example: German-*ish* cache coherence protocol

Client i:

$\texttt{Cache}[i] \in \{\mathsf{E}, \mathsf{S}, \mathsf{I}\}$

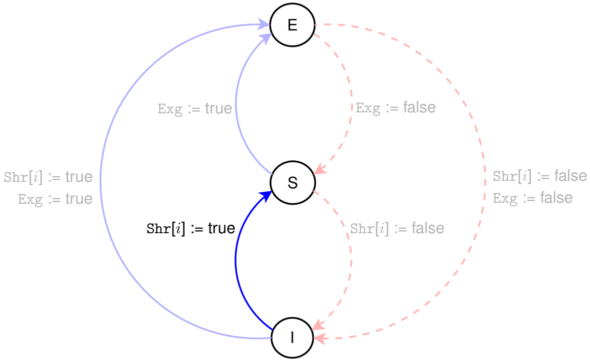
Directory:

$\texttt{Cmd} \in \{\mathsf{rs}, \mathsf{re}, \epsilon\}$

$\texttt{Ptr} \in proc$

$\texttt{Shr}[i] \in \{\mathsf{true}, \mathsf{false}\}$

$\texttt{Exg} \in \{\mathsf{true}, \mathsf{false}\}$



$$t_5 : \exists i. \quad \texttt{Ptr} = i \ \wedge \ \texttt{Cmd} = \mathsf{rs} \ \wedge \ \neg\texttt{Exg} \ \wedge$$
$$\texttt{Cmd}' = \epsilon \ \wedge \ \texttt{Shr}'[i] \ \wedge \ \texttt{Cache}'[i] = \mathsf{S}$$

BRAB algorithm

I : inital states U : unsafe states (cubes) $\mathcal{T}$: transitions

---

BRAB ():

$\mathtt{B} := \emptyset$; $\mathtt{Kind}(\mathrm{U}) := \mathrm{Orig}$; $\mathtt{From}(\mathrm{U}) := \mathrm{U}$;

$\mathcal{M} := \mathtt{FWD}(d_{max}, k)$;

while BWDA() = unsafe **do**

  if $\mathtt{Kind}(\mathtt{F})$ = Orig **then return** unsafe

  $\mathtt{B} := \mathtt{B} \cup \{ \mathtt{From}(\mathtt{F}) \}$;

return safe

BRAB algorithm

I : inital states U : unsafe states (cubes) $\mathcal{T}$: transitions

BWD ():

$\mathtt{V} := \emptyset$;

push($\mathtt{Q}$, U);

while not empty($\mathtt{Q}$) **do**

φ := pop($\mathtt{Q}$);

if $\varphi \wedge$ I sat **then return** unsafe

if $\neg(\varphi \models \bigvee_{\psi \in \mathtt{V}} \psi)$ **then**

$\mathtt{V} := \mathtt{V} \cup \{\varphi\}$;

push($\mathtt{Q}$, $\mathtt{pre}_{\mathcal{T}}(\varphi)$);

return safe

BRAB algorithm

I : inital states   U : unsafe states (cubes) $\mathcal{T}$: transitions

---

BWDA ():

$\mathtt{V} := \emptyset$;

push($\mathtt{Q}$, U);

while not empty($\mathtt{Q}$) **do**

φ := pop($\mathtt{Q}$);

if $\varphi \wedge$ I sat **then return** unsafe

if $\neg(\varphi \models \bigvee_{\psi \in \mathtt{V}} \psi)$ **then**

$\mathtt{V} := \mathtt{V} \cup \{\varphi\}$;

push($\mathtt{Q}$, $\boxed{\text{Approx}_{\mathcal{T}}(\varphi)}$);

return safe

BRAB algorithm

I : inital states U : unsafe states (cubes) $\mathcal{T}$: transitions

---

Approx$_{\mathcal{T}}$ (φ):

foreach ψ in $\texttt{candidates}(\varphi)$ **do**

if $\psi \notin \mathrm{B}$ $\wedge$ $\mathcal{M} \not\models \psi$ **then**

$\texttt{Kind}(\psi)$:= $\boxed{\text{Appr}}$;

$\ldots$

return ψ

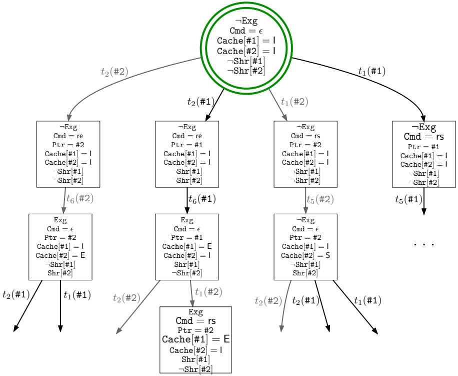
$\ldots$

return $\texttt{pre}_{\mathcal{T}}(\varphi)$

Example: BRAB on German-*ish*

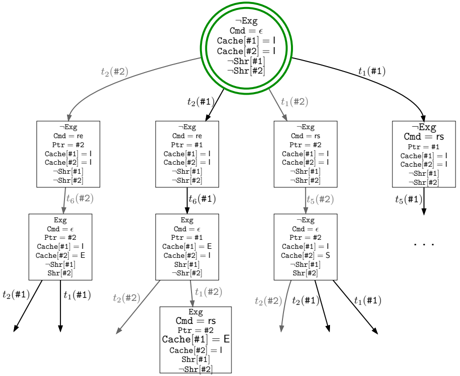
Example: BRAB on German-*ish*

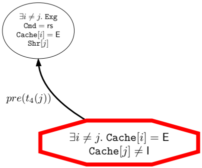
Example: BRAB on German-*ish*



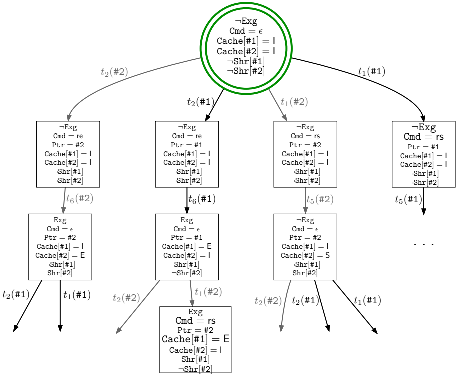
$\exists i \neq j.\ \mathtt{Cache}[i] = \mathsf{E}$
$\mathtt{Cache}[j] \neq \mathsf{I}$

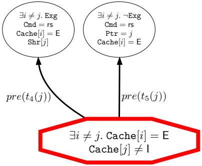
Example: BRAB on German-*ish*



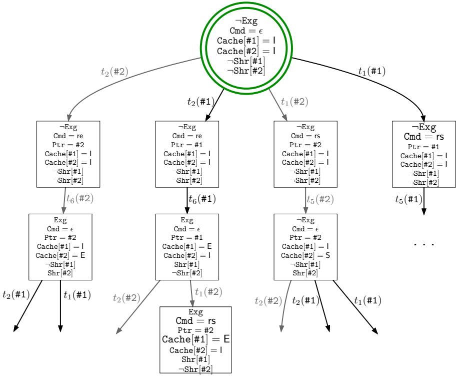


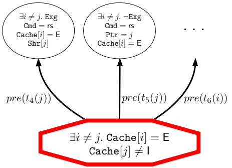
Example: BRAB on German-*ish*



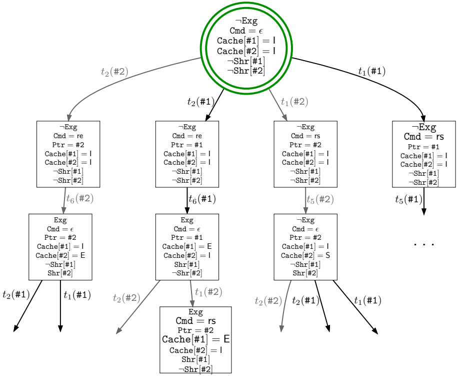


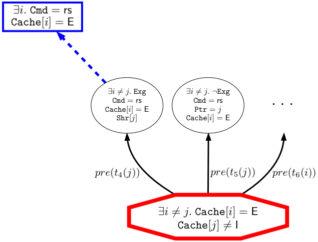
Example: BRAB on German-*ish*



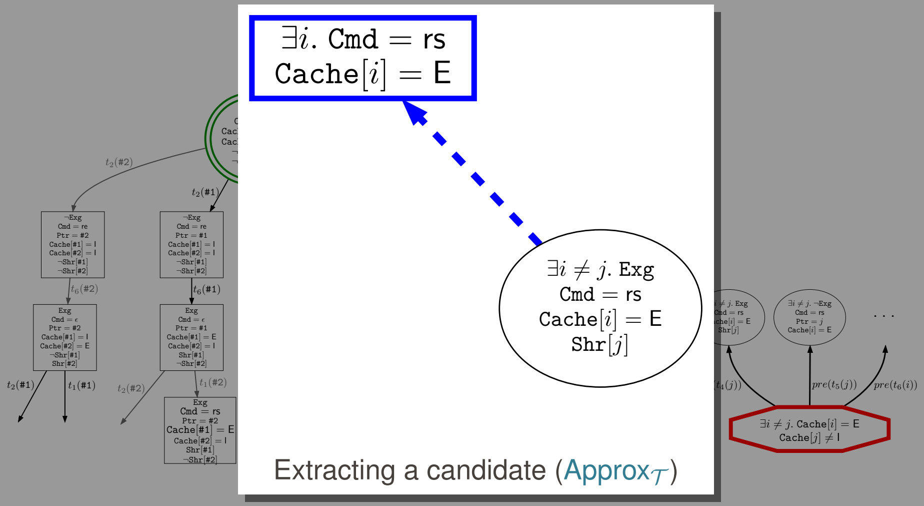


Example: BRAB on German-*ish*

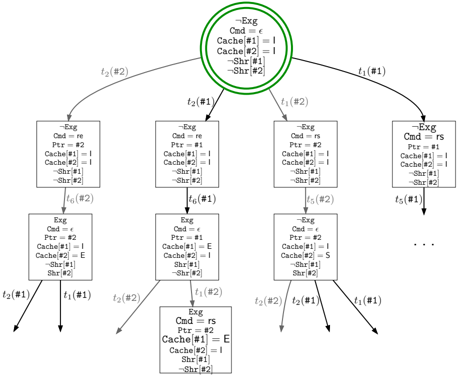


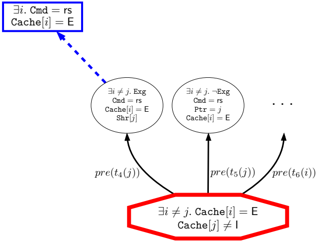


Example: BRAB on German-*ish*

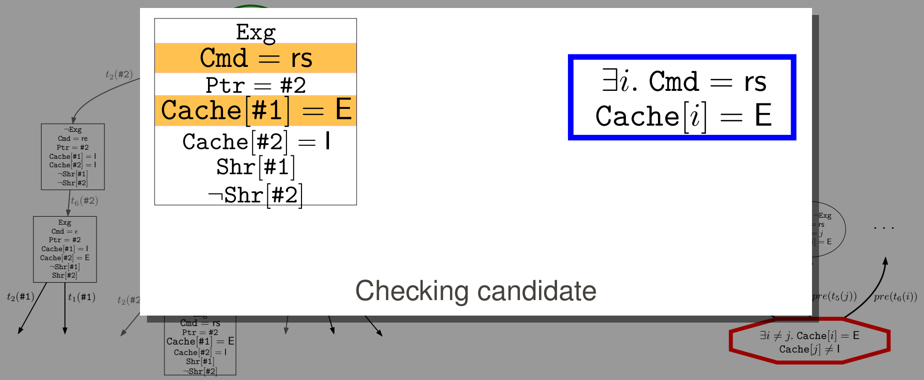


Example: BRAB on German-*ish*



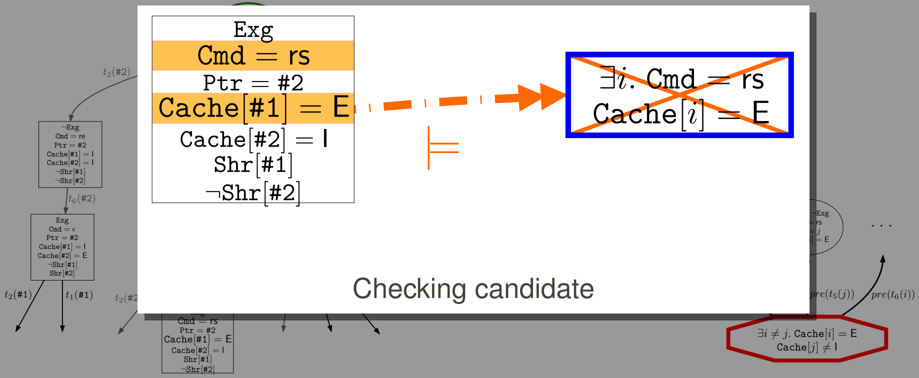


Example: BRAB on German-*ish*

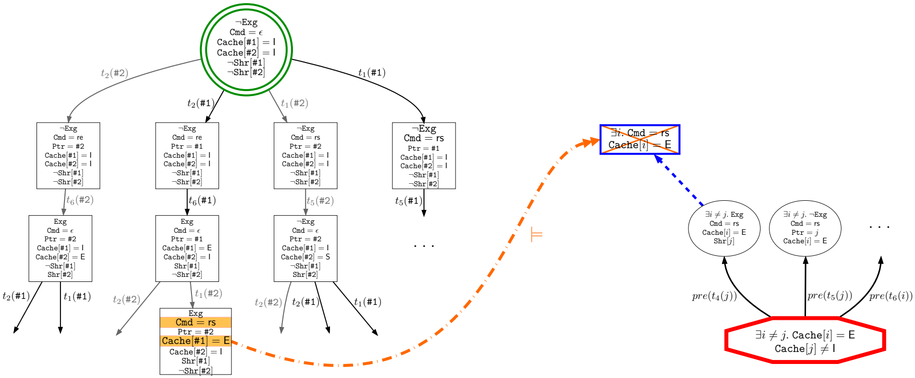


Checking candidate

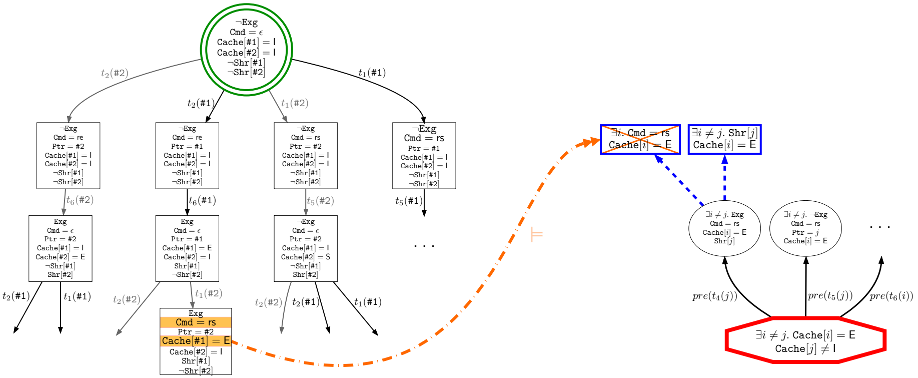
Example: BRAB on German-*ish*



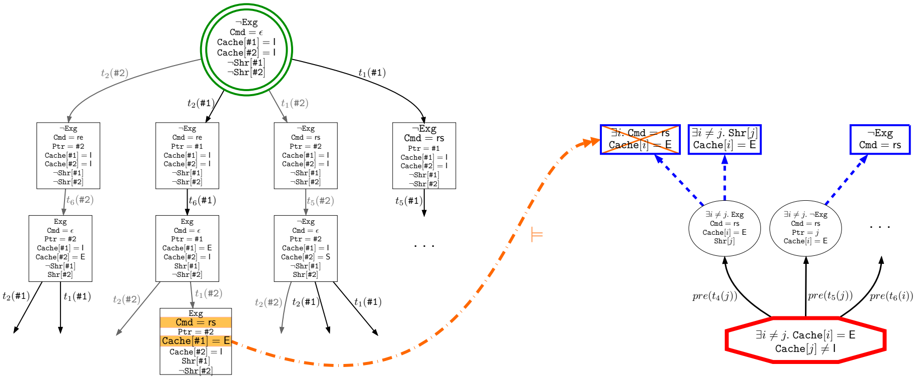
Example: BRAB on German-*ish*



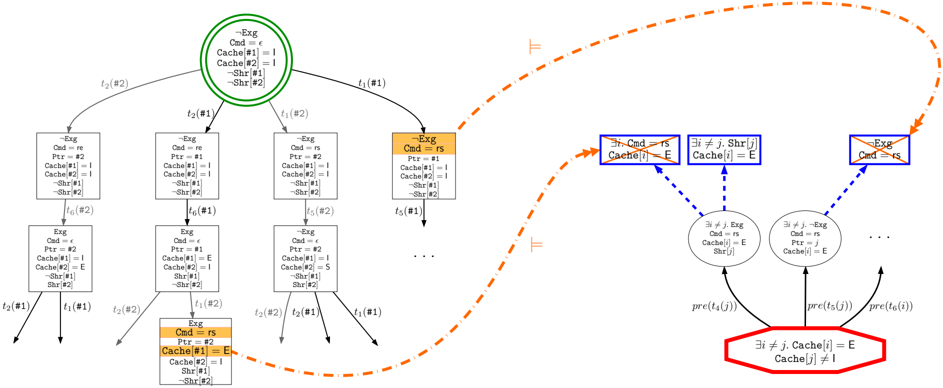
Example: BRAB on German-*ish*



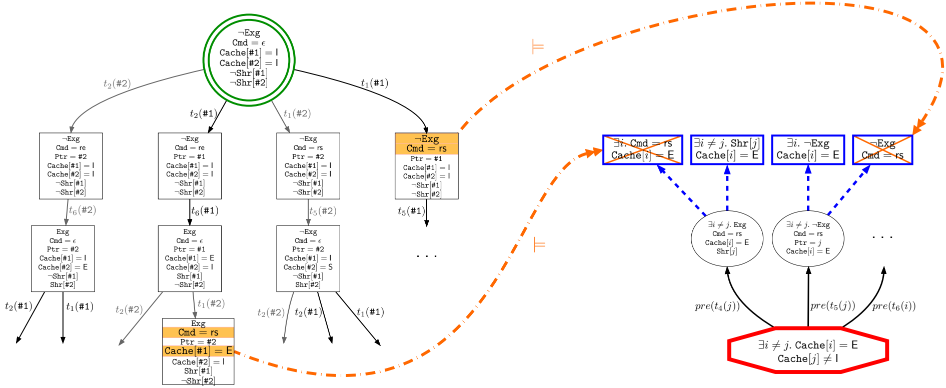
Example: BRAB on German-*ish*



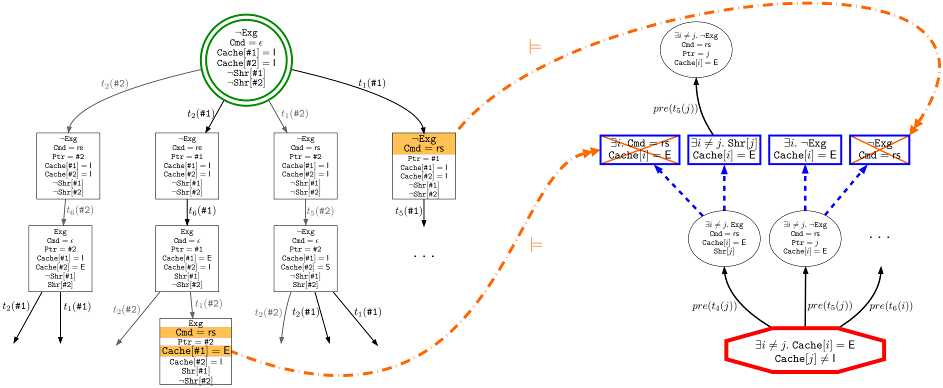
Example: BRAB on German-*ish*

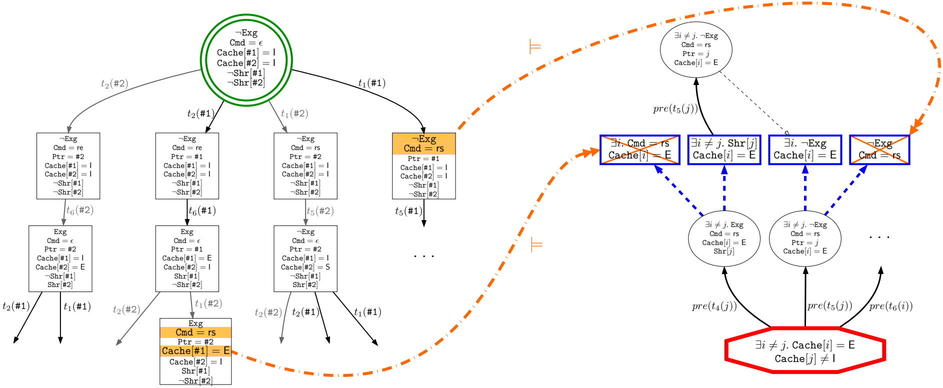


Example: BRAB on German-*ish*



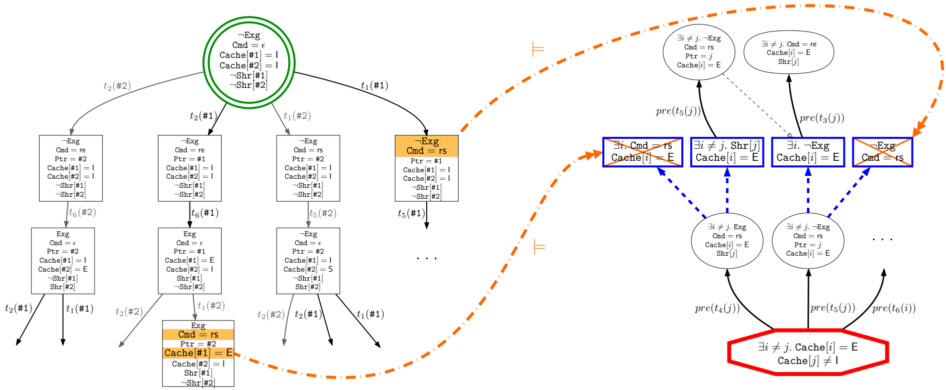
Example: BRAB on German-*ish*



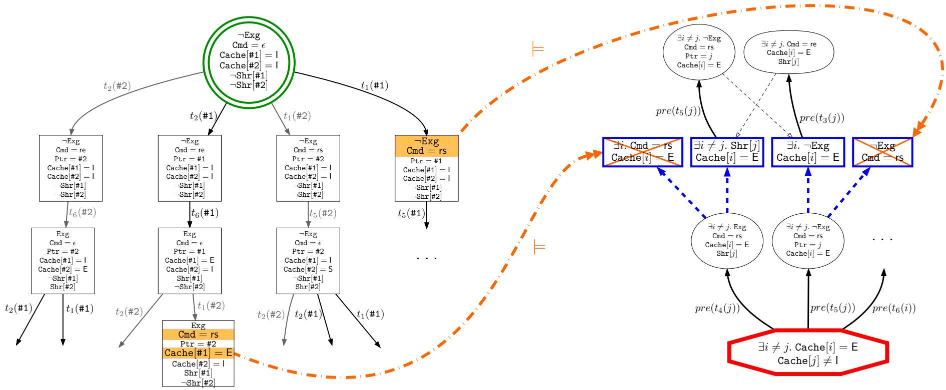


¬Exg
Cmd = ε
Cache[#1] = I
Cache[#2] = I
¬Shr[#1]
¬Shr[#2]
t2(#2)
t2(#1)
t1(#2)
t1(#1)
¬Exg
Cmd = re
Ptr = #2
Cache[#1] = I
Cache[#2] = I
¬Shr[#1]
¬Shr[#2]
¬Exg
Cmd = re
Ptr = #1
Cache[#1] = I
Cache[#2] = I
¬Shr[#1]
¬Shr[#2]
¬Exg
Cmd = rs
Ptr = #2
Cache[#1] = I
Cache[#2] = I
¬Shr[#1]
¬Shr[#2]
¬Exg
Cmd = rs
Ptr = #1
Cache[#1] = I
Cache[#2] = I
¬Shr[#1]
¬Shr[#2]
t6(#2)
t6(#1)
t5(#2)
t5(#1)
Exg
Cmd = ε
Ptr = #2
Cache[#1] = I
Cache[#2] = E
¬Shr[#1]
Shr[#2]
Exg
Cmd = ε
Ptr = #1
Cache[#1] = E
Cache[#2] = I
Shr[#1]
¬Shr[#2]
¬Exg
Cmd = ε
Ptr = #2
Cache[#1] = I
Cache[#2] = S
¬Shr[#1]
Shr[#2]
. . .
t2(#1)
t1(#1)
t2(#2)
t1(#2)
t2(#2)
t2(#1)
t1(#1)
Exg
Cmd = rs
Ptr = #2
Cache[#1] = E
Cache[#2] = I
Shr[#1]
¬Shr[#2]
⊨
⊨
∃i ≠ j. ¬Exg
Cmd = rs
Ptr = j
Cache[i] = E
pre(t5(j))
∃i. Cmd = rs
Cache[i] = E
∃i ≠ j. Shr[j]
Cache[i] = E
∃i. ¬Exg
Cache[i] = E
¬Exg
Cmd = rs
∃i ≠ j. Exg
Cmd = rs
Cache[i] = E
Shr[j]
∃i ≠ j. ¬Exg
Cmd = rs
Ptr = j
Cache[i] = E
. . .
pre(t4(j))
pre(t5(j))
pre(t6(i))
∃i ≠ j. Cache[i] = E
Cache[j] ≠ I

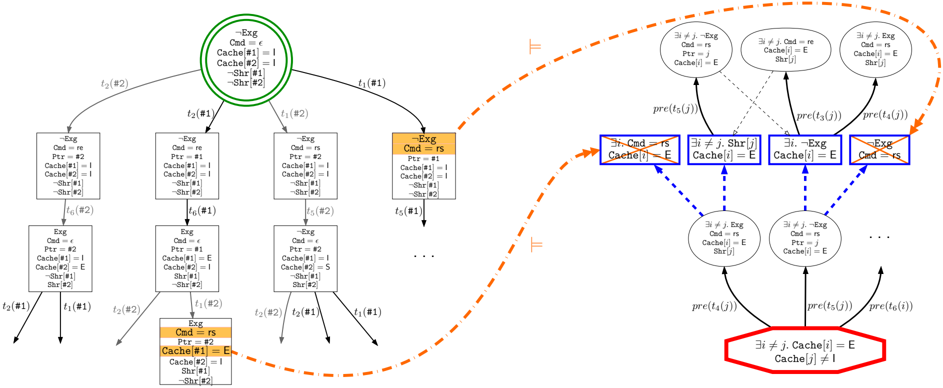
Example: BRAB on German-*ish*



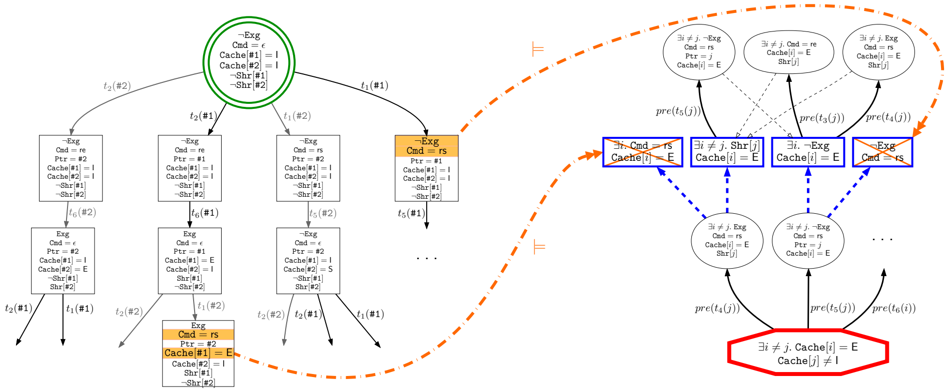
Example: BRAB on German-*ish*



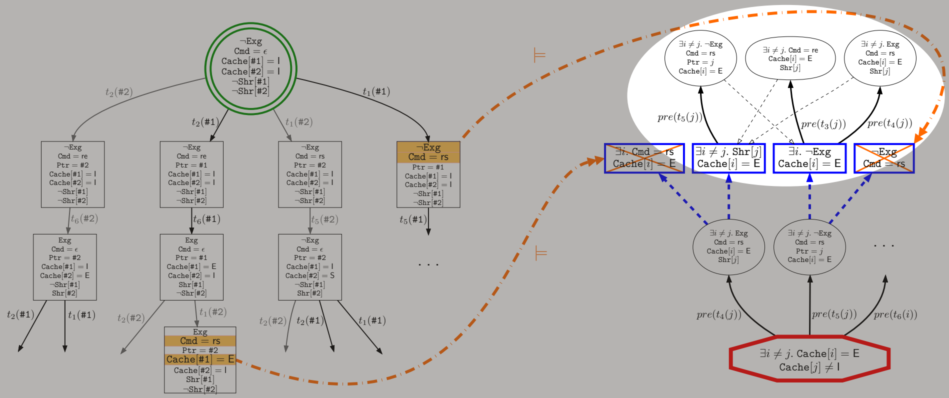
Example: BRAB on German-*ish*



Example: BRAB on German-*ish*



Example: BRAB on German-*ish*



Some benchmarks

	BRAB	Cubicle	CMurphi		
Szymanski_at	**0.14s**	0.30s	8.04s (8)	5m12s (10)	2h50m (12)
German_Baukus	**0.25s**	7.03s	0.74s (4)	19m35s (8)	4h49m (10)
German.CTC	**0.29s**	3m23s	1.83s (4)	43m46s (8)	12h35m (10)
German_pfs	**0.34s**	3m58s	0.99s (4)	22m56s (8)	5h30m (10)
Chandra-Toueg	**2m17s**	2h01m	5.68s (4)	2m58s (5)	1h36m (6)
Szymanski_na	**0.19s**	T.O.	0.88s (4)	8m25s (6)	7h08m (8)
Flash_nodata	**0.36s**	O.M.	4.86s (3)	3m33s (4)	2h46m (5)
Flash	**5m40s**	O.M.	1m27s (3)	2h15m (4)	O.M. (5)

O.M. $>$ 20 GB

T.O. $>$ 20 h

Remarks

- BRAB is complete only if the framework admits a complete Backward Reachability
- Cubicle goes beyond decidable fragment of array-based systems
- FLASH is expressed outside of this fragment
- BRAB remains safe

Future work

Improvements:

- Experiment with real size industrial protocols
- Improve backtracking
- Difficult to discover candidates for numerical invariants

Certification:

- Deductive program verification (with **Why3** and **Alt-Ergo**) and code extraction
- **Goal**: obtain a certified and efficient model checker

Thank you

Visit our web site

http://cubicle.lri.fr