SAT-Based Quantified Symmetric Minimization of the Reachable States of Distributed Protocols: An Update

Yun-Rong Lauren Luo [1]   Aman Goel [2]   Karem Sakallah [1]

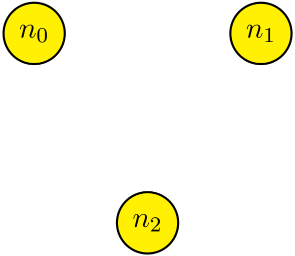
[1]University of Michigan, Ann Arbor, MI, USA

[2]Amazon Web Services, Seattle, USA

2024 International Symposium On Leveraging Applications of Formal Methods, Verification and Validation

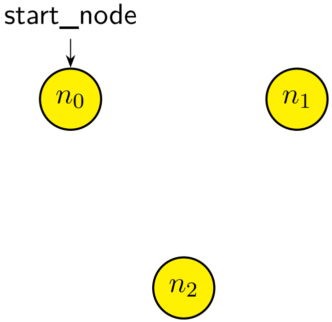
Distributed Protocol: `simple-decentralized-lock`

```
type node
```



Distributed Protocol: `simple-decentralized-lock`

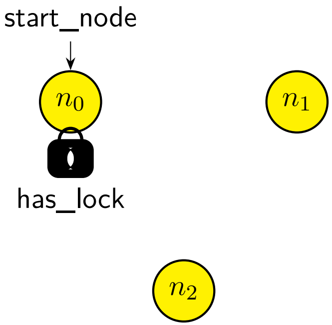
```
type node
individual start_node: node
```



Distributed Protocol: `simple-decentralized-lock`

```
type node
individual start_node: node

relation message(Src: node, Dst: node)
relation has_lock(N: node)
after init {
 message(Src, Dst) := false;
 has_lock(X) := X = start_node;
}
```

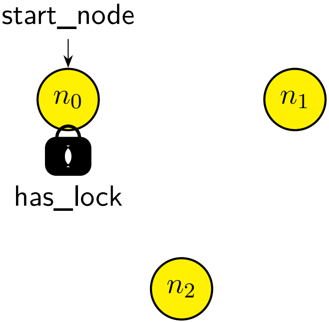


Distributed Protocol: `simple-decentralized-lock`

```
type node
individual start_node: node

relation message(Src: node, Dst: node)
relation has_lock(N: node)
after init {
 message(Src, Dst) := false;
 has_lock(X) := X = start_node;
}

action send(src: node, dst: node) = {
    assume has_lock(src);
    message(src, dst) := true;
    has_lock(src) := false;
}
```

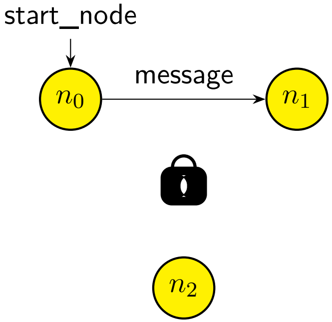


Distributed Protocol: `simple-decentralized-lock`

```
type node
individual start_node: node

relation message(Src: node, Dst: node)
relation has_lock(N: node)
after init {
 message(Src, Dst) := false;
 has_lock(X) := X = start_node;
}

action send(src: node, dst: node) = {
    assume has_lock(src);
    message(src, dst) := true;
    has_lock(src) := false;
}
```

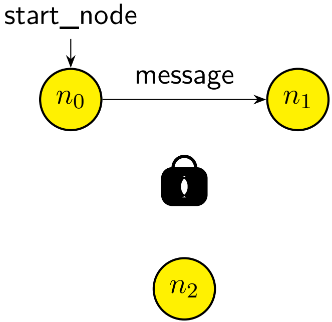


Distributed Protocol: `simple-decentralized-lock`

```
type node
individual start_node: node

relation message(Src: node, Dst: node)
relation has_lock(N: node)
after init {
 message(Src, Dst) := false;
 has_lock(X) := X = start_node;
}

action send(src: node, dst: node) = {
    assume has_lock(src);
    message(src, dst) := true;
    has_lock(src) := false;
}
action recv(src: node, dst: node) = {
    assume message(src, dst);
    message(src, dst) := false;
    has_lock(dst) := true;
}
```

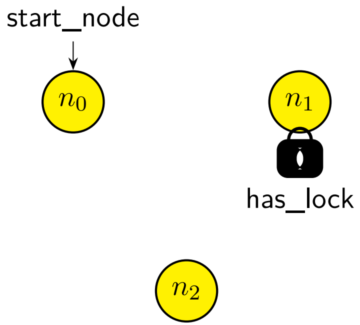


Distributed Protocol: `simple-decentralized-lock`

```
type node
individual start_node: node

relation message(Src: node, Dst: node)
relation has_lock(N: node)
after init {
 message(Src, Dst) := false;
 has_lock(X) := X = start_node;
}

action send(src: node, dst: node) = {
    assume has_lock(src);
    message(src, dst) := true;
    has_lock(src) := false;
}
action recv(src: node, dst: node) = {
    assume message(src, dst);
    message(src, dst) := false;
    has_lock(dst) := true;
}
```

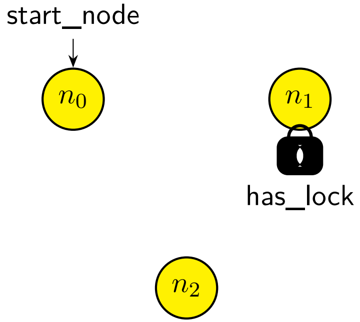


Distributed Protocol: `simple-decentralized-lock`

```
type node
individual start_node: node

relation message(Src: node, Dst: node)
relation has_lock(N: node)
after init {
 message(Src, Dst) := false;
 has_lock(X) := X = start_node;
}

action send(src: node, dst: node) = {
    assume has_lock(src);
    message(src, dst) := true;
    has_lock(src) := false;
}
action recv(src: node, dst: node) = {
    assume message(src, dst);
    message(src, dst) := false;
    has_lock(dst) := true;
}
invariant [safety] (has_lock(X) & has_lock(Y)) -> (X = Y)
```



Safety: no two nodes have lock at the same time

Previous Work [FGS23]: Deriving $R_{\min}$ for Distributed Protocols

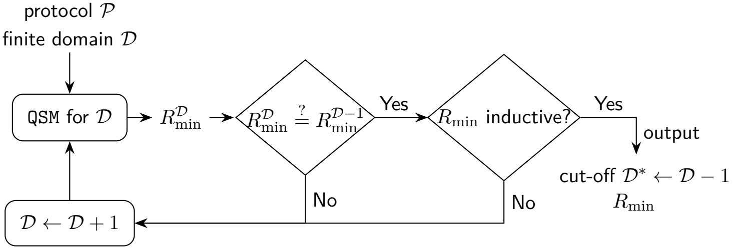
Formal Methods in Computer-Aided Design 2023

SAT-Based Quantified Symmetric Minimization of the Reachable States of Distributed Protocols

Katalin Fazekas
TU Wien
Vienna, Austria
katalin.fazekas@tuwien.ac.at

Aman Goel
Amazon Web Services§
Seattle, USA
goelaman@amazon.com

Karem A. Sakallah
University of Michigan
Ann Arbor, USA
karem@umich.edu



The output $R_{\min}$ is closed under transition for the protocol $\mathcal{P}$ of an arbitrary (potentially unbounded) domain size.

Previous $R_{\min}$ for `simp-dec-lock`

```
// Rmin
invariant [inv_0] (forall N0, N1 . ((N0 = N1) | ~has_lock(N1) | ~message(N0, N0)))
invariant [inv_1] (forall N0, N1 . ((N0 = N1) | ~message(N1, N1) | ~message(N0, N0)))
invariant [inv_2] (forall N0, N1, N2 . (~message(N0, N0)  | ~message(N1, N2)) | (N0 = N1) | (N0 = N2) | (N1 = N2) )
invariant [inv_3] (forall N0, N1 . ((N0 = N1) | ~message(N0, N1) | ~message(N0, N0)))
invariant [inv_4] (forall N0, N1 . ((N0 = N1) | ~message(N1, N0) | ~message(N0, N0)))
invariant [inv_5] (forall N0, N1 . (~(start_node = N1) | ~(start_node = N0) | (N0 = N1)))
invariant [inv_6] (forall N0, N1 . ((N0 = N1) | ~message(N0, N1) | ~has_lock(N0)))
invariant [inv_7] (forall N0, N1, N2 . ((N0 = N2) | (N1 = N2) | (N0 = N1) | ~message(N0, N2) | ~message(N1, N2)))
invariant [inv_8] (forall N0, N1, N2 . ((N0 = N2) | (N1 = N2) | ~has_lock(N1) | (N0 = N1) | ~message(N0, N2)))
invariant [inv_9] (forall N0, N1, N2 . ((N0 = N2) | (N1 = N2) | ~message(N2, N1) | (N0 = N1) | ~message(N0, N2)))
invariant [inv_10] (forall N0, N1 . ((N0 = N1) | ~has_lock(N0) | ~message(N1, N0)))
invariant [inv_11] (forall N0, N1 . ((N0 = N1) | ~message(N0, N1) | ~message(N1, N0)))
invariant [inv_12] (forall N0 . (~has_lock(N0) | ~message(N0, N0)))
invariant [inv_13] (forall N0, N1, N2 . ((N0 = N2) | (N1 = N2) | (N0 = N1) | ~message(N0, N1) | ~message(N0, N2)))
invariant [inv_14] (forall N0, N1, N2, N3 . (  ~message(N0, N3) | ~message(N1, N2)
                                              | (N0 = N2) | (N1 = N2) | (N0 = N1) | (N0 = N3) | (N1 = N3) | (N2 = N3) ))
invariant [inv_15] (forall N0, N1 . ((N0 = N1) | ~has_lock(N0) | ~has_lock(N1)))
invariant [inv_16] (exists N0 . (start_node = N0))
invariant [inv_17] (exists N0, N1, N2 . (has_lock(N0) | message(N1, N2)))
```

- The inductive invariant $R_{\min}$ for `simp-dec-lock` is a conjunction of 18 invariants.

Updated $R_{\min}$ for simp-dec-lock

```
// Rmin
invariant [inv_0] forall N0,N1. ~has_lock(N0) | ~has_lock(N1) | (N0 = N1)
invariant [inv_1] forall N0,N1,N2. ~has_lock(N0) | ~message(N1,N2)
invariant [inv_2] exists N0,N1,N2. message(N1,N2) | has_lock(N0)
invariant [inv_3] forall Src0,Src1,Dst0,Dst1. ~message(Src0,Dst0) | ~message(Src1,Dst1)
                                              | (Src0 = Src1 & Dst0 = Dst1)
```

- $R_{\min}$ reduces from 18 invariants to 4 invariants.
- The derivation of the former $R_{\min}$ only considers symmetric minimization whereas the updated $R_{\min}$ considers both symmetric and logical minimization.

Proving Safety Property P with $R_{\min}$

- The derivation of $R_{\min}$ is independent of the safety properties hence $R_{\min}$ can be used to prove any safety property P.
- Given a safety property P, P holds for the protocol if $R_{\min} \rightarrow P \equiv \top$, i.e., $R_{\min} \wedge \neg P \equiv \bot$.
- An example safety property:

```
// safety property P
invariant [safety] (has_lock(X) & has_lock(Y)) -> (X = Y)

// Rmin
invariant [inv_0] forall N0,N1. ~has_lock(N0) | ~has_lock(N1) | (N0 = N1)
invariant [inv_1] forall N0,N1,N2. ~has_lock(N0) | ~message(N1,N2)
invariant [inv_2] exists N0,N1,N2. message(N1,N2) | has_lock(N0)
invariant [inv_3] forall Src0,Src1,Dst0,Dst1. ~message(Src0,Dst0) | ~message(Src1,Dst1)
                                              | (Src0 = Src1 & Dst0 = Dst1)
```

Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)

protocol $\mathcal{P}$
finite domain $\mathcal{D}$

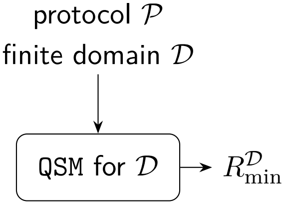
Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)

protocol $\mathcal{P}$
finite domain $\mathcal{D}$

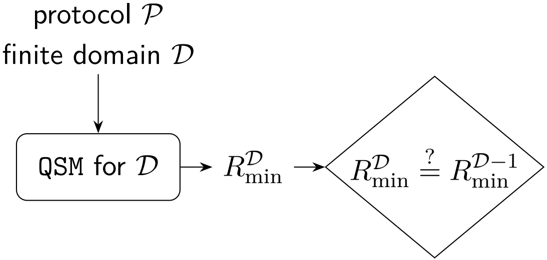
$\downarrow$

`QSM` for $\mathcal{D}$

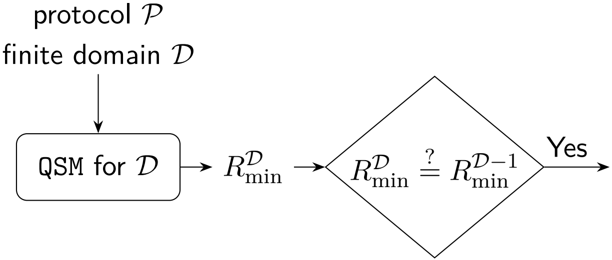
Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)



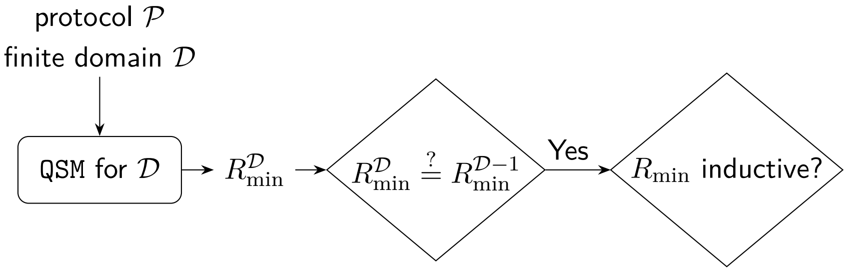
Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)



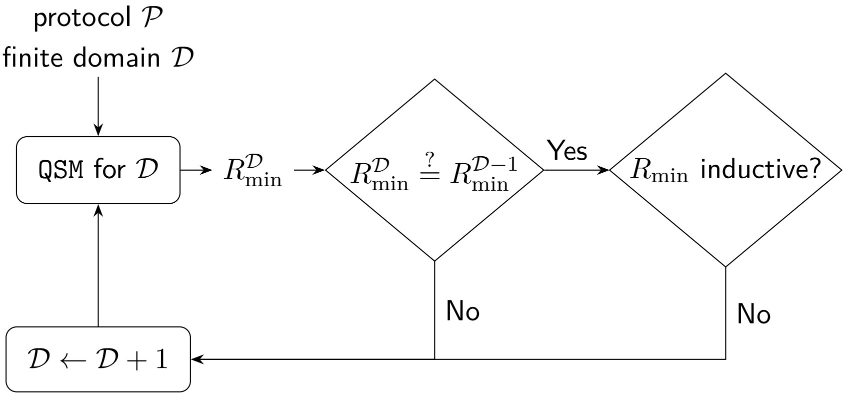
Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)



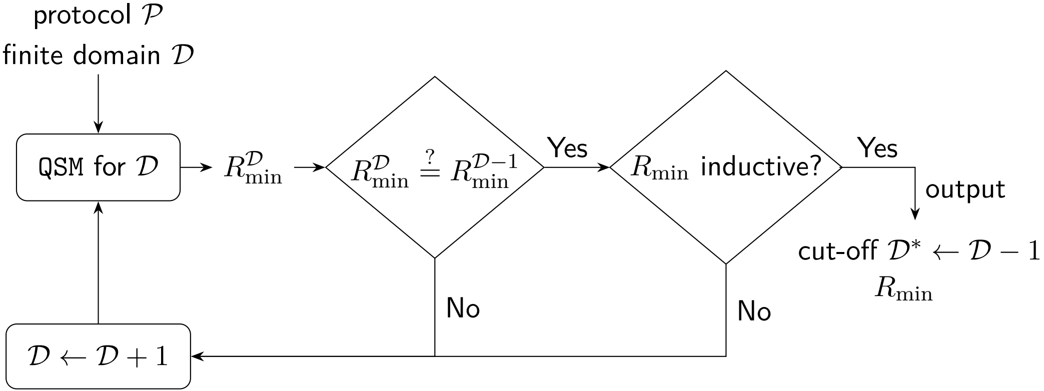
Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)



Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)



Deriving $R_{\min}$: Quantified Symmetric Minimization (QSM)



Deriving $R^{\mathcal{D}}_{\min}$: Quantified Symmetric Minimization (QSM)

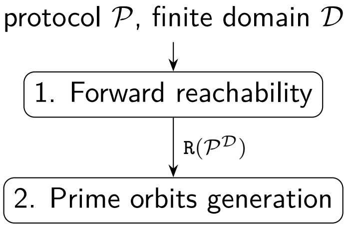
protocol $\mathcal{P}$, finite domain $\mathcal{D}$

Deriving $R_{\min}^{\mathcal{D}}$: Quantified Symmetric Minimization (QSM)

$$\text{protocol } \mathcal{P}\text{, finite domain } \mathcal{D} \downarrow \boxed{\text{1. Forward reachability}}$$

1. Use BDD-based symbolic image computation to derive reachable states $\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.

Deriving $R^{\mathcal{D}}_{\min}$: Quantified Symmetric Minimization (QSM)



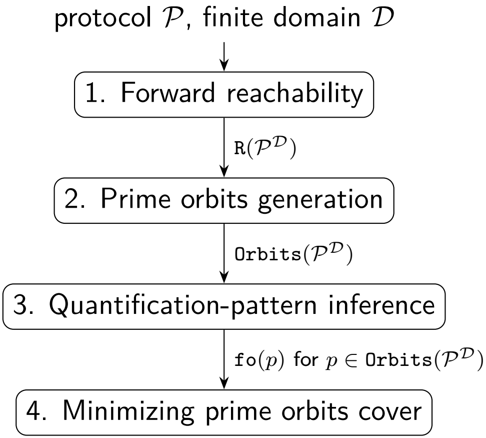
1. Use BDD-based symbolic image computation to derive reachable states $\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.
2. Use a SAT-based method to enumerate symmetric prime orbits $\mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$ for the unreachable states $\neg\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.

Deriving $R^{\mathcal{D}}_{\min}$: Quantified Symmetric Minimization (QSM)

$$
\begin{tikzcd}
\text{protocol } \mathcal{P}\text{, finite domain } \mathcal{D} \arrow[d] \\
\boxed{\text{1. Forward reachability}} \arrow[d, "\mathtt{R}(\mathcal{P}^{\mathcal{D}})"] \\
\boxed{\text{2. Prime orbits generation}} \arrow[d, "\mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})"] \\
\boxed{\text{3. Quantification-pattern inference}}
\end{tikzcd}
$$

1. Use BDD-based symbolic image computation to derive reachable states $\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.
2. Use a SAT-based method to enumerate symmetric prime orbits $\mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$ for the unreachable states $\neg\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.
3. Express each prime orbit $p \in \mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$ with a logically equivalent first-order sentence $\mathtt{fo}(p)$.

Deriving $R^{\mathcal{D}}_{\min}$: Quantified Symmetric Minimization (QSM)



1. Use BDD-based symbolic image computation to derive reachable states $\texttt{R}(\mathcal{P}^{\mathcal{D}})$.
2. Use a SAT-based method to enumerate symmetric prime orbits $\texttt{Orbits}(\mathcal{P}^{\mathcal{D}})$ for the unreachable states $\neg\texttt{R}(\mathcal{P}^{\mathcal{D}})$.
3. Express each prime orbit $p \in \texttt{Orbits}(\mathcal{P}^{\mathcal{D}})$ with a logically equivalent first-order sentence $\texttt{fo}(p)$.
4. Use a SAT-based branch-and-bound search for a minimum-cost subset $\texttt{MinOrbits}(\mathcal{P}^{\mathcal{D}})$ of $\texttt{Orbits}(\mathcal{P}^{\mathcal{D}})$ that forms a set cover for $\neg\texttt{R}(\mathcal{P}^{\mathcal{D}})$.

Deriving $R^{\mathcal{D}}_{\min}$: Quantified Symmetric Minimization (QSM)

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

$\downarrow$

1. Forward reachability

$\downarrow$ $\mathtt{R}(\mathcal{P}^{\mathcal{D}})$

2. Prime orbits generation

$\downarrow$ $\mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$

3. Quantification-pattern inference

$\downarrow$ $\mathtt{fo}(p)$ for $p \in \mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$

4. Minimizing prime orbits cover

$\downarrow$ $\mathtt{MinOrbits}(\mathcal{P}^{\mathcal{D}})$

finite reachability $R^{\mathcal{D}}_{\min}$

1. Use BDD-based symbolic image computation to derive reachable states $\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.
2. Use a SAT-based method to enumerate symmetric prime orbits $\mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$ for the unreachable states $\neg\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.
3. Express each prime orbit $p \in \mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$ with a logically equivalent first-order sentence $\mathtt{fo}(p)$.
4. Use a SAT-based branch-and-bound search for a minimum-cost subset $\mathtt{MinOrbits}(\mathcal{P}^{\mathcal{D}})$ of $\mathtt{Orbits}(\mathcal{P}^{\mathcal{D}})$ that forms a set cover for $\neg\mathtt{R}(\mathcal{P}^{\mathcal{D}})$.
5. Construct $R^{\mathcal{D}}_{\min} := \bigwedge_{p \in \mathtt{MinOrbits}(\mathcal{P}^{\mathcal{D}})} \neg\mathtt{fo}(p)$

Quantified Symmetric Minimization (QSM): Issues and Updates

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

↓

1. Forward reachability

↓

2. Prime orbits generation

↓

3. Quantification-pattern inference

↓

4. Minimizing prime orbits cover

↓

finite reachability $R^{\mathcal{D}}_{\min}$

Quantified Symmetric Minimization (QSM): Issues and Updates

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

$\downarrow$

1. Forward reachability

$\downarrow$

2. Prime orbits generation

$\downarrow$

3. Quantification-pattern inference

$\downarrow$

4. Minimizing prime orbits cover

$\downarrow$

finite reachability $R^{\mathcal{D}}_{\min}$

Issue I: BDD-based symbolic image computation is unscalable.

Quantified Symmetric Minimization (QSM): Issues and Updates

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

↓

1. Forward reachability

↓

2. Prime orbits generation

↓

3. Quantification-pattern inference

↓

4. Minimizing prime orbits cover

↓

finite reachability $R^{\mathcal{D}}_{\min}$

Issue I: BDD-based symbolic image computation is unscalable.

Update I: Replace BDD-based method with Symmetric Quotient Depth-First Search (`Sym_DFS`).

Quantified Symmetric Minimization (QSM): Issues and Updates

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

1. Forward reachability

2. Prime orbits generation

3. Quantification-pattern inference

4. Minimizing prime orbits cover

finite reachability $R^{\mathcal{D}}_{\min}$

Issue I: BDD-based symbolic image computation is unscalable.

Update I: Replace BDD-based method with Symmetric Quotient Depth-First Search (`Sym_DFS`).

Issue II: The symmetric prime orbits are not necessarily logically prime.

Quantified Symmetric Minimization (QSM): Issues and Updates

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

↓

1. Forward reachability

↓

2. Prime orbits generation

↓

3. Quantification-pattern inference

↓

4. Minimizing prime orbits cover

↓

finite reachability $R_{\min}^{\mathcal{D}}$

Issue I: BDD-based symbolic image computation is unscalable.

Update I: Replace BDD-based method with Symmetric Quotient Depth-First Search (`Sym_DFS`).

Issue II: The symmetric prime orbits are not necessarily logically prime.

Update II: Merge symmetric prime orbits into a logically equivalent prime *super orbit*.

Quantified Symmetric Minimization (QSM): Issues and Updates

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

↓

1. Forward reachability

↓

2. Prime orbits generation

↓

3. Quantification-pattern inference

↓

4. Minimizing prime orbits cover

↓

finite reachability $R^{\mathcal{D}}_{\min}$

Issue I: BDD-based symbolic image computation is unscalable.

Update I: Replace BDD-based method with Symmetric Quotient Depth-First Search (`Sym_DFS`).

Issue II: The symmetric prime orbits are not necessarily logically prime.

Update II: Merge symmetric prime orbits into a logically equivalent prime *super orbit*.

Issue III: There exists multiple minimum solutions at small domains $\mathcal{D}$.

Quantified Symmetric Minimization (QSM): Issues and Updates

protocol $\mathcal{P}$, finite domain $\mathcal{D}$

↓

1. Forward reachability

↓

2. Prime orbits generation

↓

3. Quantification-pattern inference

↓

4. Minimizing prime orbits cover

↓

finite reachability $R_{\min}^{\mathcal{D}}$

Issue I: BDD-based symbolic image computation is unscalable.

Update I: Replace BDD-based method with Symmetric Quotient Depth-First Search (`Sym_DFS`).

Issue II: The symmetric prime orbits are not necessarily logically prime.

Update II: Merge symmetric prime orbits into a logically equivalent prime *super orbit*.

Issue III: There exists multiple minimum solutions at small domains $\mathcal{D}$.

Update III: Update the syntactic convergence condition to the smallest $\mathcal{D}^*$ such that $R_{\min}^{\mathcal{D}^*}$ is unique.

Update I: Symmetric Quotient DFS for Reachable States (`Sym_DFS`)

- Issues of BDD-based symbolic image computation:

Update I: Symmetric Quotient DFS for Reachable States (`Sym_DFS`)

- Issues of BDD-based symbolic image computation:
  - Struggles when the BDD that represents the reachable states is large.

Update I: Symmetric Quotient DFS for Reachable States (`Sym_DFS`)

- Issues of BDD-based symbolic image computation:
  - Struggles when the BDD that represents the reachable states is large.
  - Does not leverage the structural symmetry of protocols.

Update I: Symmetric Quotient DFS for Reachable States (`Sym_DFS`)

- Issues of BDD-based symbolic image computation:
    - Struggles when the BDD that represents the reachable states is large.
    - Does not leverage the structural symmetry of protocols.

- `Sym_DFS`:

Update I: Symmetric Quotient DFS for Reachable States (`Sym_DFS`)

- Issues of BDD-based symbolic image computation:
  - Struggles when the BDD that represents the reachable states is large.
  - Does not leverage the structural symmetry of protocols.

- `Sym_DFS`:

Update I: Symmetric Quotient DFS for Reachable States (Sym_DFS)

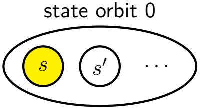
- Issues of BDD-based symbolic image computation:
  - Struggles when the BDD that represents the reachable states is large.
  - Does not leverage the structural symmetry of protocols.

- Sym_DFS:
  - Upon discovering a new state s,

Update I: Symmetric Quotient DFS for Reachable States (Sym_DFS)

- Issues of BDD-based symbolic image computation:
  - Struggles when the BDD that represents the reachable states is large.
  - Does not leverage the structural symmetry of protocols.

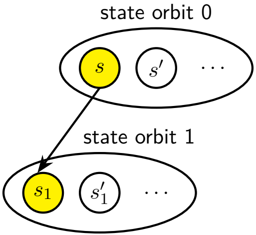
- Sym_DFS:
  - Upon discovering a new state s, Sym_DFS enumerates the symmetric orbit of s and assigns s to be a representative state.



Update I: Symmetric Quotient DFS for Reachable States (`Sym_DFS`)

- Issues of BDD-based symbolic image computation:
  - Struggles when the BDD that represents the reachable states is large.
  - Does not leverage the structural symmetry of protocols.

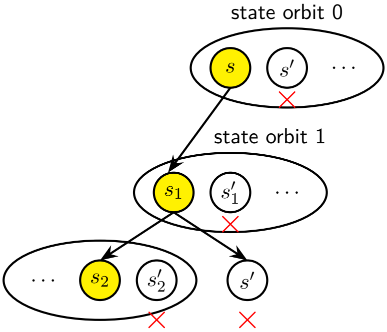
- `Sym_DFS`:
  - Upon discovering a new state s, `Sym_DFS` enumerates the symmetric orbit of s and assigns s to be a representative state.
  - `Sym_DFS` only explores successors for a representative state s.



Update I: Symmetric Quotient DFS for Reachable States (`Sym_DFS`)

- Issues of BDD-based symbolic image computation:
  - Struggles when the BDD that represents the reachable states is large.
  - Does not leverage the structural symmetry of protocols.

- `Sym_DFS`:
  - Upon discovering a new state s, `Sym_DFS` enumerates the symmetric orbit of s and assigns s to be a representative state.
  - `Sym_DFS` only explores successors for a representative state s.
  - `Sym_DFS` does not explore successors for a non-representative state s'.



Update I: BDD vs. Sym_DFS

		DFS		BDD	Time (s)		Peak Mem (MB)	
Protocol	#vars	#states	repr%	#cubes	DFS	BDD	DFS	BDD
sharded_kv	84	24389	0.46%	13824	146	**33**	15.3	19.3
sharded_kv_no_lost_keys	84	21952	0.41%	13824	117	**33**	14.0	18.9
toy_consensus	40	693	2.16%	?	**14**	TO	1.7	-
toy_consensus_epr	40	693	2.16%	?	**14**	TO	1.7	-
naive_consensus	36	873	2.75%	432	18	**2**	1.7	4.8
simple-election	44	2552	1.92%	2425	**32**	2,058	2.4	4.8
toy_consensus_forall	49	235875	0.04%	?	**4,872**	TO	198.2	-
consensus_epr	72	?	?	1318	TO	**1,202**	-	5.9

TO stands for time out after 5000 sec.

Update I: BDD vs. Sym_DFS

		DFS		BDD	Time (s)		Peak Mem (MB)	
Protocol	#vars	#states	repr%	#cubes	DFS	BDD	DFS	BDD
sharded_kv	84	24389	0.46%	13824	146	**33**	15.3	19.3
sharded_kv_no_lost_keys	84	21952	0.41%	13824	117	**33**	14.0	18.9
toy_consensus	40	693	2.16%	?	**14**	TO	1.7	-
toy_consensus_epr	40	693	2.16%	?	**14**	TO	1.7	-
naive_consensus	36	873	2.75%	432	18	**2**	1.7	4.8
simple-election	44	2552	1.92%	2425	**32**	2,058	2.4	4.8
toy_consensus_forall	49	235875	0.04%	?	**4,872**	TO	198.2	-
consensus_epr	72	?	?	1318	TO	**1,202**	-	5.9

TO stands for time out after 5000 sec.

Update I: BDD vs. Sym_DFS

		DFS		BDD	Time (s)		Peak Mem (MB)	
Protocol	#vars	#states	repr%	#cubes	DFS	BDD	DFS	BDD
sharded_kv	84	24389	0.46%	13824	146	**33**	15.3	19.3
sharded_kv_no_lost_keys	84	21952	0.41%	13824	117	**33**	14.0	18.9
toy_consensus	40	693	2.16%	?	**14**	TO	1.7	-
toy_consensus_epr	40	693	2.16%	?	**14**	TO	1.7	-
naive_consensus	36	873	2.75%	432	18	**2**	1.7	4.8
simple-election	44	2552	1.92%	2425	**32**	2,058	2.4	4.8
toy_consensus_forall	49	235875	0.04%	?	**4,872**	TO	198.2	-
consensus_epr	72	?	?	1318	TO	**1,202**	-	5.9

TO stands for time out after 5000 sec.

Update I: BDD vs. `Sym_DFS`

		DFS		BDD	Time (s)		Peak Mem (MB)	
Protocol	#vars	#states	repr%	#cubes	DFS	BDD	DFS	BDD
sharded_kv	84	24389	0.46%	13824	146	**33**	15.3	19.3
sharded_kv_no_lost_keys	84	21952	0.41%	13824	117	**33**	14.0	18.9
toy_consensus	40	693	2.16%	?	**14**	TO	1.7	-
toy_consensus_epr	40	693	2.16%	?	**14**	TO	1.7	-
naive_consensus	36	873	2.75%	432	18	**2**	1.7	4.8
simple-election	44	2552	1.92%	2425	**32**	2,058	2.4	4.8
toy_consensus_forall	49	235875	0.04%	?	**4,872**	TO	198.2	-
consensus_epr	72	?	?	1318	TO	**1,202**	-	5.9

TO stands for time out after 5000 sec.

Update II: Quantification-Pattern Inference for Super Orbits

5 orbits for `simp-dec-lock` at $|\texttt{node}| = 3$

orbit representative cube : orbit size

$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_0, n_0) : 3$

$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_0, n_1) : 6$

$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_0) : 6$

$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_1) : 6$

$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_2) : 6$

Update II: Quantification-Pattern Inference for Super Orbits

5 orbits for `simp-dec-lock` at $|\texttt{node}| = 3$

orbit representative cube : orbit size	Logically equivalent first-order sentence for each orbit
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_0, n_0) : 3$	$\exists N_0.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_0, N_0)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_0, n_1) : 6$	$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_0, N_1) \wedge (N_0 \neq N_1)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_0) : 6$	$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_1, N_0) \wedge (N_0 \neq N_1)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_1) : 6$	$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_1, N_1) \wedge (N_0 \neq N_1)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_2) : 6$	$\exists N_0, N_1, N_2.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_1, N_2) \wedge (N_0 \neq N_1) \wedge (N_0 \neq N_2) \wedge (N_1 \neq N_2)$

Update II: Quantification-Pattern Inference for Super Orbits

Super orbit

5 orbits for simp-dec-lock at $|\texttt{node}| = 3$

orbit representative cube : orbit size

orbit representative cube : orbit size	Logically equivalent first-order sentence for each orbit
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_0, n_0) : 3$	$\exists N_0.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_0, N_0)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_0, n_1) : 6$	$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_0, N_1) \wedge (N_0 \neq N_1)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_0) : 6$	$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_1, N_0) \wedge (N_0 \neq N_1)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_1) : 6$	$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_1, N_1) \wedge (N_0 \neq N_1)$
$\texttt{has\_lock}(n_0) \wedge \texttt{message}(n_1, n_2) : 6$	$\exists N_0, N_1, N_2.\ \texttt{has\_lock}(N_0) \wedge \texttt{message}(N_1, N_2) \wedge (N_0 \neq N_1) \wedge (N_0 \neq N_2) \wedge (N_1 \neq N_2)$

Update II: Quantification-Pattern Inference for Super Orbits

Super orbit

5 orbits for `simp-dec-lock` at $|\texttt{node}| = 3$

orbit representative cube : orbit size

$\texttt{has\_lock}(n_0) \land \texttt{message}(n_0, n_0) : 3$

$\texttt{has\_lock}(n_0) \land \texttt{message}(n_0, n_1) : 6$

$\texttt{has\_lock}(n_0) \land \texttt{message}(n_1, n_0) : 6$

$\texttt{has\_lock}(n_0) \land \texttt{message}(n_1, n_1) : 6$

$\texttt{has\_lock}(n_0) \land \texttt{message}(n_1, n_2) : 6$

Logically equivalent first-order sentence for each orbit

$\exists N_0.\ \texttt{has\_lock}(N_0) \land \texttt{message}(N_0, N_0)$

$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \land \texttt{message}(N_0, N_1) \land (N_0 \neq N_1)$

$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \land \texttt{message}(N_1, N_0) \land (N_0 \neq N_1)$

$\exists N_0, N_1.\ \texttt{has\_lock}(N_0) \land \texttt{message}(N_1, N_1) \land (N_0 \neq N_1)$

$\exists N_0, N_1, N_2.\ \texttt{has\_lock}(N_0) \land \texttt{message}(N_1, N_2) \land (N_0 \neq N_1) \land (N_0 \neq N_2) \land (N_1 \neq N_2)$

$\exists N_0, N_1, N_2.\ \texttt{has\_lock}(N_0) \land \neg\texttt{message}(N_1, N_2)$

Logically equivalent first-order sentence for super orbit

Update II: Quantification-Pattern Inference for Super Orbits

Perform quantification-pattern inference on super orbits:

- Yields a smaller number of quantified invariants in $R_{\min}$.
  - 10 out of 16 protocols results in a more compact $R_{\min}$.
  - E.g., the number of invariants in $R_{\min}$ for `simp-dec-lock` reduces from 18 to 4.
- Reaches cut-off at a smaller domain $\mathcal{D}^*$.
  - 8 out of 16 protocols achieves a smaller cut-off domain size.
  - E.g., the cut-off domain size for `simp-dec-lock` reduces from $|\texttt{node}| = 4$ to $|\texttt{node}| = 3$.

Update III: Syntactic Convergence Condition

- Protocol `toy_consensus_forall`:
  - There are two minimum solutions for $R^{\mathcal{D}}_{\min}$ when the domain size for `node` is 3 or 4.
  - When the domain size is $|\texttt{node}| = 5, |\texttt{quorum}| = 10, |\texttt{value}| = 3$, Solution 1 is the only minimum solution.

```
// Solution 0 for Rmin
invariant [inv_19] forall N. (exists V. vote(N,V) | ~voted(N))
invariant [inv_2] forall V,N. voted(N) | ~vote(N,V)
invariant [inv_3] forall V0,V1,N. ~vote(N,V0) | ~vote(N,V1) | (V0 = V1)
invariant [inv_14] forall V,N,Q. ~decided(V) | vote(N,V) | member(N,Q) | voting_quorum = Q

// Solution 1 for Rmin
invariant [inv_19] forall N. (exists V. vote(N,V) | ~voted(N))
invariant [inv_2] forall V,N. voted(N) | ~vote(N,V)
invariant [inv_3] forall V0,V1,N. ~vote(N0,V0) | ~vote(N0,V1) | (V0 = V1)
invariant [inv_12] forall V,N,Q. ~decided(V) | vote(N,V) | ~member(N,Q) | voting_quorum ~= Q
```

Update III: Syntactic Convergence Condition

- Protocol `toy_consensus_forall`:
  - There are two minimum solutions for $R^{\mathcal{D}}_{\min}$ when the domain size for `node` is 3 or 4.
  - When the domain size is $|\texttt{node}| = 5, |\texttt{quorum}| = 10, |\texttt{value}| = 3$, Solution 1 is the only minimum solution.

```
// Solution 0 for Rmin
invariant [inv_19] forall N. (exists V. vote(N,V) | ~voted(N))
invariant [inv_2] forall V,N. voted(N) | ~vote(N,V)
invariant [inv_3] forall V0,V1,N. ~vote(N,V0) | ~vote(N,V1) | (V0 = V1)
invariant [inv_14] forall V,N,Q. ~decided(V) | vote(N,V) | member(N,Q) | voting_quorum = Q

// Solution 1 for Rmin
invariant [inv_19] forall N. (exists V. vote(N,V) | ~voted(N))
invariant [inv_2] forall V,N. voted(N) | ~vote(N,V)
invariant [inv_3] forall V0,V1,N. ~vote(N0,V0) | ~vote(N0,V1) | (V0 = V1)
invariant [inv_12] forall V,N,Q. ~decided(V) | vote(N,V) | ~member(N,Q) | voting_quorum ~= Q
```

- Let the syntactic convergence condition be the smallest domain $\mathcal{D}^*$ such that (1) $R^{\mathcal{D}^*}_{\min}$ is unique; (2) $R^{\mathcal{D}^*}_{\min} = R^{\mathcal{D}^*+1}_{\min}$.

Overall Results on 19 Protocols

Protocol	cut-off	#inv	#inv [FGS23]	Time (s)
Consensus	value=3	**1**	1	8.78
TCommit	resource_manager=2	**7**	8	8.79
Ricart-Agrawala	node=2	**4**	4	8.86
lock_server	server=1,client=3	**3**	3	8.40
sharded_kv	node=3,key=2,value=2	**5**	8	10.09
sharded_kv_no_lost_key	node=3,key=2,value=2	**6**	9	10.71
simple-decentralized-lock	node=3	**4**	18	9.00
firewall	node=3	**5**	5	11.73
lockserv	node=3	**10**	13	8.95
lockserv_automaton	node=3	**10**	13	9.09
client_server_ae	node=1,request=1,response=1	**3**	?	8.80
TwoPhase	resource_manager=1	**16**	?	50.07
toy_consensus	node=3,quorum=3,value=3	**4**	4	9.90
toy_consensus_epr	node=3,quorum=3,value=3	**5**	6	10.20
naive_consensus	node=3,quorum=3,value=3	**3**	3	10.17
simple-election	acceptor=3,quorum=3,proposer=3	**5**	7	13.25
toy_consensus_forall	node=5,quorum=10,value=3	**4**	6	172.03
consensus_epr	node=3,quorum=3,value=2	**10**	16	755.09
quorum-leader-election-wo-maj	node=5,nset=10	**5**	?	390.93

Future Work

- `Sym_DFS`:

- Quantification-pattern inference:

- Branch-and-bound search for minimum prime orbits cover

- Totally-ordered sorts:

Future Work

- `Sym_DFS`:
  - Issue: enumerating the symmetric orbit of a newly discovered state can be inefficient when the symmetric group is large.
- Quantification-pattern inference:
- Branch-and-bound search for minimum prime orbits cover
- Totally-ordered sorts:

Future Work

- `Sym_DFS`:
  - Issue: enumerating the symmetric orbit of a newly discovered state can be inefficient when the symmetric group is large.
  - Future work: develop a hashing technique that hashes symmetric states to the same hash key.
- Quantification-pattern inference:
- Branch-and-bound search for minimum prime orbits cover
- Totally-ordered sorts:

Future Work

- `Sym_DFS`:
  - Issue: enumerating the symmetric orbit of a newly discovered state can be inefficient when the symmetric group is large.
  - Future work: develop a hashing technique that hashes symmetric states to the same hash key.
- Quantification-pattern inference:
  - For protocol `two_phase_commit`, there is no bounded quantification pattern to capture the special literal distribution in prime orbits: $\#\text{alive} + \#\text{decide\_abort} = |\text{node}|$
- Branch-and-bound search for minimum prime orbits cover
- Totally-ordered sorts:

Future Work

- `Sym_DFS`:
  - Issue: enumerating the symmetric orbit of a newly discovered state can be inefficient when the symmetric group is large.
  - Future work: develop a hashing technique that hashes symmetric states to the same hash key.
- Quantification-pattern inference:
  - For protocol `two_phase_commit`, there is no bounded quantification pattern to capture the special literal distribution in prime orbits: $\#\text{alive} + \#\text{decide\_abort} = |\text{node}|$
  - These prime orbits can be viewed as having an infinite cost and excluded from the minimum solution. Future work: how to identify them?
- Branch-and-bound search for minimum prime orbits cover
- Totally-ordered sorts:

Future Work

- `Sym_DFS`:
  - Issue: enumerating the symmetric orbit of a newly discovered state can be inefficient when the symmetric group is large.
  - Future work: develop a hashing technique that hashes symmetric states to the same hash key.
- Quantification-pattern inference:
  - For protocol `two_phase_commit`, there is no bounded quantification pattern to capture the special literal distribution in prime orbits: $\#\text{alive} + \#\text{decide\_abort} = |\text{node}|$
  - These prime orbits can be viewed as having an infinite cost and excluded from the minimum solution. Future work: how to identify them?
- Branch-and-bound search for minimum prime orbits cover
  - `QSM` uses a prime's coverage as the branching heuristic, which is *estimated* using SAT queries.
- Totally-ordered sorts:

Future Work

- `Sym_DFS`:
  - Issue: enumerating the symmetric orbit of a newly discovered state can be inefficient when the symmetric group is large.
  - Future work: develop a hashing technique that hashes symmetric states to the same hash key.
- Quantification-pattern inference:
  - For protocol `two_phase_commit`, there is no bounded quantification pattern to capture the special literal distribution in prime orbits: $\#\text{alive} + \#\text{decide\_abort} = |\text{node}|$
  - These prime orbits can be viewed as having an infinite cost and excluded from the minimum solution. Future work: how to identify them?
- Branch-and-bound search for minimum prime orbits cover
  - `QSM` uses a prime's coverage as the branching heuristic, which is *estimated* using SAT queries.
  - Future work: improve coverage estimation with approximate or exact #SAT queries.
- Totally-ordered sorts:

Future Work

- `Sym_DFS`:
  - Issue: enumerating the symmetric orbit of a newly discovered state can be inefficient when the symmetric group is large.
  - Future work: develop a hashing technique that hashes symmetric states to the same hash key.
- Quantification-pattern inference:
  - For protocol `two_phase_commit`, there is no bounded quantification pattern to capture the special literal distribution in prime orbits: $\#\text{alive} + \#\text{decide\_abort} = |\text{node}|$
  - These prime orbits can be viewed as having an infinite cost and excluded from the minimum solution. Future work: how to identify them?
- Branch-and-bound search for minimum prime orbits cover
  - `QSM` uses a prime's coverage as the branching heuristic, which is *estimated* using SAT queries.
  - Future work: improve coverage estimation with approximate or exact #SAT queries.
- Totally-ordered sorts:
  - Extend `QSM` to support protocols with totally-ordered sorts by exploiting *temporal repetitiveness*.

Conclusions

- `QSM`: Behaviors of unbounded distributed protocols can be inferred from analyzing finite instances until a small *cut-off* domain sizes for protocol.

Conclusions

- `QSM`: Behaviors of unbounded distributed protocols can be inferred from analyzing finite instances until a small *cut-off* domain sizes for protocol.
- Updates to `QSM`:

Conclusions

- `QSM`: Behaviors of unbounded distributed protocols can be inferred from analyzing finite instances until a small *cut-off* domain sizes for protocol.
- Updates to `QSM`:
  - We propose `Sym_DFS`, quantification-pattern inference for super orbits, and updated syntactic convergence condition.

Conclusions

- `QSM`: Behaviors of unbounded distributed protocols can be inferred from analyzing finite instances until a small *cut-off* domain sizes for protocol.
- Updates to `QSM`:
  - We propose `Sym_DFS`, quantification-pattern inference for super orbits, and updated syntactic convergence condition.
  - Experiments show that these improvements allow `QSM` to solve previously unsolved cases, produce more compact quantified inductive invariants, and achieve syntactic convergence at smaller domain sizes.

References I

[FGS23] Katalin Fazekas, Aman Goel, and Karem A. Sakallah. "SAT-Based Quantified Symmetric Minimization of the Reachable States of Distributed Protocols". In: *Formal Methods in Computer-Aided Design (FMCAD 2023)*. Ames, Iowa, Oct. 2023, pp. 152–161. DOI: 10.34727/2023/isbn.978-3-85448-060-0_23.