Cardinalities and Universal Quantifiers for Verifying Parameterized Systems

Klaus v. Gleissenthall, UC San Diego and TU Munich

Nikolaj Bjørner and Andrey Rybalchenko, Microsoft Research
Parallel / Distributed systems

- Paxos/Chubby
- Zab/Zookeeper
- Raft

Verify!
Consensus: majorities

That means $v = w$
One third rule:

Nodes try to agree on a value

Broadcast value in each round

Update own value

Messages may be lost
One third rule:

- received 1,3,3
- choose 3
- n = # of nodes
- Hear form >2/3n then choose most received
- >2/3n share value v then decide on v
One third rule:

Received 3,3,3

3

3

3

decide on value 3

Hear form $>\frac{2}{3}n$

then

choose most received

$>\frac{2}{3}n$ share value $v$

then
decide on $v$
One third rule: property

forall p,p’: decide(p,v) and decide(p’,w) then v=w
One third rule: invariant

forall p: decide(p,v) then
#{t | candidate(t)=v } > 2/3n

- Quantification: number of processes not known statically
- Agreement by: no two majorities
- Count # of nodes with same candidate
Filter lock: description

- n threads in total
- start at level 0
- critical section: level n-1
- try to increase level

level n-1

...
Filter lock: description

Advance *iff*

no one is above you

someone else is at your level
Filter lock: description

- no one is above you
- someone else is at your level

Levels:
- Level 0
- Level 1
- Level 2
Filter Lock: property

\[ \# \{ t \mid \text{lv}(t) = n-1 \} \leq 1 \]

Mutual exclusion
Filter Lock: invariant

\[ \forall l: 0 \leq l \leq n-1 \rightarrow \#\{ t \mid \text{lv}(t) \geq l \} \leq n-l \]
Filter lock:

- Level 0
- Level 1
- ... (n-1 levels)
- Level n-1

- 1 thread
- ... (n-1 threads)
- n threads

Preserved under:
- No one is above you
- Someone else is at your level

Mutual exclusion
Program Verifiers:

IntegersControlFlow

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Property</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Property</td>
<td>Proof</td>
</tr>
</tbody>
</table>

IntegersControlFlow

Arrays

1. ESBMC
2. SMACK+Corral
3. Symbiotic

IntegersControlFlow

Arrays

1. ESBMC
2. SMACK+Corral
3. Symbiotic
In this talk:

1: What to count
2: How to do the counting

#π

Verifies properties show before

Sharpie
A simple example:

Some number \( n \) of threads

If there is a thread at location 2, then \( a > 0 \)

```
global int a=0;
1:   a++;
2:
```
Example: in logic

**Initial states:**
\[ \forall t: pc(t) = 1 \land a = 0 \]

**Transition relation:**
\[ pc(me) = 1 \land pc'(me \leftarrow 2) \land a' = a + 1 \]

**Safety:**
\[ pc(me) = 2 \rightarrow a > 0 \]

Local variables as functions

Primed = after transition
Example: constraints

find: \textit{inv}

$\forall t: pc(t)=1 \land a=0 \rightarrow \textit{inv}(a,pc)$

$\textit{inv}(a,pc) \land pc(me)=1 \land pc':=pc(me\leftarrow 2) \land a'=a+1 \rightarrow \textit{inv}(a',pc')$

$\textit{inv}(a,pc) \rightarrow pc(me)=2 \rightarrow a>0$

Horn clauses
Solving:

\[\#\{t \mid pc(t) > 1\} \leq a\]
How to count: invariant checking

∀t: pc(t)=1

#\{t \mid pc(t) > 1\} \leq a \quad \rightarrow \quad pc(me)=2 \rightarrow a>0

Checking: not supported by existing method
How to count: invariant checking

\[ \forall t: pc(t)=1 \land a=0 \rightarrow 0 \leq 0 \]

Set is empty

\[ 1 \leq a \rightarrow pc(me)=2 \rightarrow a > 0 \]

Set is non-empty
Example: point wise update

#\{t \mid pc(t)>1\} \leq a \land pc'(me) = pc(me \leftarrow 2) \land a' = a+1 \rightarrow #\{t \mid pc'(t)>1\} \leq a'

How to deal with upd?
Example: point wise update

\[ pc'(t) > 1 \]

\[ pc(me) = 1 \land pc' := pc(me \leftarrow 2) \]
What to count

Find $s$ and $\text{inv}$

Cardinality free

$\# \text{inv} \cdot \land \text{inv}(a, pc, k)$

$\#\pi$ chooses $s$
s.t. the proof goes through
Cardinality axioms:
Example: finding the solution

\[ \#\{t \mid s(t)\} = k \]

\[ \forall t: pc(t) = 1 \land a = 0 \]

\[ (\forall t: \neg s(t)) \rightarrow k = 0 \]

\[ \text{inv}(a, pc, k) \]
Cardinality axioms:

\[ s = \text{pc}(t) > 1 \]

\[ \text{inv} = k = \text{<a} \]
### Evaluation:

<table>
<thead>
<tr>
<th>Program</th>
<th>Card</th>
<th>Property</th>
<th>Inferred cardinalities</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>intro [21]</td>
<td>✓</td>
<td>$(\exists t : pc(t) = 2) \rightarrow b &lt; a$</td>
<td>${ t</td>
<td>pc(t) = 2 }$</td>
</tr>
<tr>
<td>bluetooth [21]</td>
<td>✓</td>
<td>$(\exists t : pc(t) = 2) \rightarrow st = 0$</td>
<td>${ t</td>
<td>pc(t) = 2 }$</td>
</tr>
<tr>
<td>tree traverse [21]</td>
<td></td>
<td></td>
<td>$\text{nodes} + 1$</td>
<td>4.2s</td>
</tr>
<tr>
<td>cache [59]</td>
<td></td>
<td>$#{ t</td>
<td>pc(t) = 3 } \leq 1$</td>
<td>$#{ t</td>
</tr>
<tr>
<td>garbage collection</td>
<td>✓</td>
<td>$#{ t</td>
<td>2 \leq pc(t) \leq 4 } \leq 1 \land m = 1$</td>
<td>$#{ t</td>
</tr>
</tbody>
</table>

- Works on problems from the literature
- First automated cardinality proofs
- Cache coherence
- Garbage collection
- Locking
- Consensus
Conclusion:

**Verifying protocols requires cardinalities**

**Sharpie**

**Knows how to count**

**Chooses what to count to complete proof**

**Existing verifiers can't count**

#π
That’s it!
Axioms: inequality

\((\forall t: s(t) \to p(t)) \to k \leq l\)
Axioms: inequality

\[ \#\{t \mid s(t)\} = k \]
\[ \#\{t \mid p(t)\} = l \]

\[(\forall t: s(t) \rightarrow p(t)) \rightarrow k \leq l\]
Axioms: strict inequality

\[
\begin{align*}
\{t \mid s\} &= k \\
\{t \mid p\} &= l \\
\{t \mid p\} &= l \\
(\forall t : s(t) \rightarrow p(t)) \land (\exists t : \neg s(t) \land p(t)) & \rightarrow k < l
\end{align*}
\]

Additional witness