ADJUSTABLE BLOCK ANALYSIS

ACTOR-BASED CREATION OF CODE BLOCK SUMMARIES FOR SCALING FORMAL VERIFICATION

Matthias Kettl
24.02.2022
1 Motivation

- The primary goal is the reduction of the needed wall time for verification.
- The decomposition of the program in code blocks allows the distribution of the verification to many workers.
- Implementation of an easily extensible framework for distributed analyses.
2 Actor Model Overview

- One actor broadcasts information to all actors.
- Communication over messages in the JSON format.
- Every worker processes every message.
- The actors react to every message differently.
2 Actor Model

Messages are four-tuples: (*type*, *id*, *target node*, *payload*).

Messages have one of five types.

A message contains information about the origin of the message and arbitrary information in the payload:

<table>
<thead>
<tr>
<th>Type</th>
<th>ID</th>
<th>TargetNode</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error, ErrorCondition,</td>
<td>The ID of the worker, where</td>
<td>The node number of the CFANode,</td>
<td>A JSON string containing arbitrary key-value</td>
</tr>
<tr>
<td>ErrorConditionUnreachable,</td>
<td>the message originated from.</td>
<td>where the message originated from.</td>
<td>pairs.</td>
</tr>
<tr>
<td>BlockPostCondition or FoundResult</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2 Actor Model

Worker

Routine

while (!finished) {
    Message m = nextMessage();  //blocks
    // may modify variable finished
    Set<Message> responses = processMessage(m);
    broadcast(responses);
}

Purpose

- Workers are the entities of our actor model.
- Messages are the unit for exchanging data.
- Workers process messages and produce a possibly empty set of messages as response.
3 Distributed Verification

Code Blocks

CFA
3 Distributed Verification

Code Blocks

CFA

Code Blocks
3 Distributed Verification

CFA

Code Blocks

Code Blocks
3 Distributed Verification

CFA

Code Blocks
3 Distributed Verification

CFA

Code Blocks

1

2

3

4

5

6

7

8

9

1

2

3

4

5

6

7

8

9

11
3 Distributed Verification

CFA

Code Blocks

Code Blocks
3 Distributed Verification

CFA

Code Blocks
3 Distributed Verification

Code Blocks

CFA

Code Blocks

Analysis Workers
3 Distributed Verification

- DCPAs extend known CPAs $\mathcal{C}$ (abstract domain, transfer relation, merge, stop) with four operators.
- DCPAs run on code blocks.
- The four operators are defined as follows:
  - **proceed**: $\mathcal{M} \mapsto B \times 2^\mathcal{M}$
  - **combine**: $\mathcal{A} \times \mathcal{A} \mapsto \mathcal{A}$
  - **serialize**: $\mathcal{A} \mapsto \mathcal{M}$
  - **deserialize**: $\mathcal{M} \mapsto \mathcal{A}$
- DCPAs support forward and backward analyses.
- DCPAs stop whenever they reach the block end.

\[\mathcal{M}\text{ the set of all possible messages}\]
\[\mathcal{A}\text{ the set of abstract states}\]
\[B\text{ Boolean values \{true, false\}}\]
3 Distributed Verification

Analysis Worker

Simplified scheme of an Analysis Worker.
3 Distributed Verification

Example

```c
int main() {
    int x = 0;
    x = x + 1;
    if (x == 1) {
        x = x + 1;
    } else {
        x = x - 1;
    }
    assert(x == 0);
}
```

Program

CFA

Block Graph
### 3 Distributed Verification Example

<table>
<thead>
<tr>
<th>Message</th>
<th>W0</th>
<th>W1</th>
<th>W2</th>
<th>W3</th>
<th>W4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>BPC</td>
<td>BPC</td>
<td>BPC</td>
<td>BPC</td>
<td>EC</td>
</tr>
<tr>
<td>WorkerID</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Node</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Payload</td>
<td>( p_{\text{CPA}}: x_0 = 0 \land x_1 = x_0 + 1 )</td>
<td>( p_{\text{CPA}}: x_0 \neq 1 \land x_1 = x_0 - 1 )</td>
<td>( p_{\text{CPA}}: x_0 = 1 \land x_1 = x_0 + 1 )</td>
<td>( p_{\text{CPA}}: x_0 = 0 )</td>
<td>( p_{\text{CPA}}: x_0 \neq 0 )</td>
</tr>
</tbody>
</table>

*Initial messages of all 5 workers*
3 Distributed Verification

**Example**

<table>
<thead>
<tr>
<th>Type</th>
<th>BPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>0</td>
</tr>
<tr>
<td>Node</td>
<td>2</td>
</tr>
<tr>
<td>Payload</td>
<td>$x_0 = 0 \land x_1 = x_0 + 1$</td>
</tr>
</tbody>
</table>

Worker 1

<table>
<thead>
<tr>
<th>Type</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>4</td>
</tr>
<tr>
<td>Node</td>
<td>5</td>
</tr>
<tr>
<td>Payload</td>
<td>pCPA: $x_0 \neq 0$</td>
</tr>
</tbody>
</table>

Diagram:

- B0: $x = 0$, $x = x + 1$
  - B1: $[x \neq 1]$, $x = x - 1$
  - B3: $[x = 0]$, safe
  - B4: $[x \neq 0]$, error
- B2: $[x = 1]$, $x = x + 1$
3 Distributed Verification

Example

<table>
<thead>
<tr>
<th>Type</th>
<th>BPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>0</td>
</tr>
<tr>
<td>Node</td>
<td>2</td>
</tr>
<tr>
<td>Payload</td>
<td>$x_0 = 0 \land x_1 = x_0 + 1$</td>
</tr>
</tbody>
</table>

Worker 1

---

ForwardAnalysis:

- $x_0 = 0 \land x_1 = x_0 + 1 \land x_1 \neq 1 \land x_2 = x_1 - 1$

---

serialize:

- \{BPC, 1, 5, pCPA:
  - $x_0 = 0 \land x_1 = x_0 + 1 \land x_1 \neq 1 \land x_2 = x_1 - 1$\}
Distributed Verification Example

Worker 1

<table>
<thead>
<tr>
<th>Type</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>4</td>
</tr>
<tr>
<td>Node</td>
<td>5</td>
</tr>
<tr>
<td>Payload</td>
<td>$p_{CPA}$: ( x_0 \neq 0 )</td>
</tr>
</tbody>
</table>

serialize:
\{ECU, 1, 5, *\}

BackwardAnalysis:
skipped

proceed:
Message targets final location? Error condition is reachable?

\[\begin{align*}
B0 &: x = 0 \\
&: x = x + 1 \\
B1 &: [x \neq 1] \\
&: x = x - 1 \\
B2 &: [x = 1] \\
&: x = x + 1 \\
B3 &: [x = 0] \\
& \text{safe} \\
B4 &: [x \neq 0] \\
& \text{error}
\end{align*}\]
3 Distributed Verification Example

<table>
<thead>
<tr>
<th>Type</th>
<th>BPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>0</td>
</tr>
<tr>
<td>Node</td>
<td>2</td>
</tr>
<tr>
<td>Payload</td>
<td>$x_0 = 0 \land x_1 = x_0 + 1$</td>
</tr>
</tbody>
</table>

Worker 2

<table>
<thead>
<tr>
<th>Type</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>4</td>
</tr>
<tr>
<td>Node</td>
<td>5</td>
</tr>
<tr>
<td>Payload</td>
<td>pCPA: $x_0 \neq 0$</td>
</tr>
</tbody>
</table>

B0: $x = 0$
B1: $x = x + 1$
B2: $[x = 1]
B3: $[x = 0]$
B4: $[x \neq 0]$
### 3 Distributed Verification Example

<table>
<thead>
<tr>
<th>Type</th>
<th>BPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>0</td>
</tr>
<tr>
<td>Node</td>
<td>2</td>
</tr>
<tr>
<td>Payload</td>
<td>$x_0 = 0 \land x_1 = x_0 + 1$</td>
</tr>
</tbody>
</table>

Worker 2

---

**Forward Analysis:**

$$
\begin{align*}
    x_0 &= 0 \land x_1 = x_0 + 1 \land x_1 = 1 \land x_2 = x_1 + 1
\end{align*}
$$

**Serialize:**

$$
\{\text{BPC, 2, 5, pCPA:} \ x_0 = 0 \land x_1 = x_0 + 1 \land x_1 = 1 \land x_2 = x_1 + 1\}$$

---

<table>
<thead>
<tr>
<th>Type</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>4</td>
</tr>
<tr>
<td>Node</td>
<td>5</td>
</tr>
<tr>
<td>Payload</td>
<td>pCPA: $x_0 \neq 0$</td>
</tr>
</tbody>
</table>

---

### B0

- $x = 0$
- $x = x + 1$

### B1

- $[x \neq 1]$
- $x = x - 1$

### B2

- $[x = 1]$
- $x = x + 1$

### B3

- $[x = 0]$
  - safe

### B4

- $[x \neq 0]$
  - error
3 Distributed Verification Example

Worker 2

<table>
<thead>
<tr>
<th>Type</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>4</td>
</tr>
<tr>
<td>Node</td>
<td>5</td>
</tr>
<tr>
<td>Payload</td>
<td>pCPA: (x_0 \neq 0)</td>
</tr>
</tbody>
</table>

serialize:
\[
\{EC, 2, 2, pCPA: \quad x_0 \neq 0 \wedge x_0 = x_1 + 1 \wedge x_1 = 1\}
\]

BackwardAnalysis:
\[
x_0 \neq 0 \wedge x_0 = x_1 + 1 \wedge x_1 = 1
\]

proceed:
Message targets final location? Error condition is reachable?

\[
\begin{align*}
x &= 0 \\
x &= x + 1
\end{align*}
\]

\[
\begin{align*}
[x \neq 1] \\
x &= x - 1
\end{align*}
\]

\[
\begin{align*}
[x = 1] \\
x &= x + 1
\end{align*}
\]

\[
\begin{align*}
[x = 0] \\
\text{safe}
\end{align*}
\]

\[
\begin{align*}
[x \neq 0] \\
\text{error}
\end{align*}
\]
3 Distributed Verification Example

SAT-check: true
{FoundResult, 0, 0, result: False}

BackwardAnalysis:
\[ x_0 \neq 0 \land x_0 = x_1 + 1 \land x_1 = 1 \land x_1 = x_2 + 1 \land x_2 = 0 \]

proceed:
Message targets final location?
Error condition is reachable?

Worker 0

<table>
<thead>
<tr>
<th>Type</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkerID</td>
<td>2</td>
</tr>
<tr>
<td>Node</td>
<td>2</td>
</tr>
</tbody>
</table>

Payload: pCPA:
\[ x_0 \neq 0 \land x_0 = x_1 + 1 \land x_1 = 1 \]

B0
\[ x = 0 \]
\[ x = x - 1 \]

B1
\[ x \neq 1 \]
\[ x = x + 1 \]

B2
\[ x = 1 \]
\[ x = x + 1 \]

B3
\[ x = 0 \]
\[ safe \]

B4
\[ x \neq 0 \]
\[ error \]

Worker 0 Type EC WorkerID 2 Node 2 Payload pCPA:
\[ x_0 \neq 0 \land x_0 = x_1 + 1 \land x_1 = 1 \]
4 Evaluation Setup

- 6671 tasks from the SV-COMP ReachSafety benchmark set.
- Benchmarks are run on two setups:
  - Setup 1: Intel Core i7-6700 @ 3.40 GHz, 8 cores, 33 GB RAM
  - Setup 2: Intel Core i7-10700 @ 2.90 GHz, 16 cores, 67 GB RAM
4 Evaluation  Soundness

- The distributed approach causes more timeouts and out-of-memory errors.
- The backward analysis causes “recursion” exceptions even if there are none.
- The verification results of the predicate analysis (if present) match the results of the distributed approach.
- Reduction of blocks/workers saves resources but can also increase the needed time for SAT-checks.

<table>
<thead>
<tr>
<th>status</th>
<th>predicate</th>
<th>DCPA_D</th>
<th>DCPA_D↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>out of memory</td>
<td>39</td>
<td>1417</td>
<td>2108</td>
</tr>
<tr>
<td>timeout</td>
<td>1920</td>
<td>2907</td>
<td>2001</td>
</tr>
<tr>
<td>error</td>
<td>1539</td>
<td>1586</td>
<td>1935</td>
</tr>
<tr>
<td>True</td>
<td>2086</td>
<td>504</td>
<td>382</td>
</tr>
<tr>
<td>False</td>
<td>1087</td>
<td>257</td>
<td>245</td>
</tr>
<tr>
<td>∑</td>
<td>6671</td>
<td>6671</td>
<td>6671</td>
</tr>
</tbody>
</table>
4 Evaluation Soundness

- The distributed approach causes more timeouts and out-of-memory errors.
- The backward analysis causes “recursion” exceptions even if there are none.
- The verification results of the predicate analysis (if present) match the results of the distributed approach.
- Reduction of blocks/workers saves resources but can also increase the needed time for SAT-checks.

<table>
<thead>
<tr>
<th></th>
<th>result</th>
<th>predicate</th>
<th>$\text{DCPA}_D$</th>
<th>$\text{DCPA}_{\downarrow}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>correct</td>
<td>2993</td>
<td>675</td>
<td>548</td>
<td></td>
</tr>
<tr>
<td>incorrect</td>
<td>180</td>
<td>86</td>
<td>79</td>
<td></td>
</tr>
</tbody>
</table>
4 Evaluation Distributing the Analysis

- DCPA(SB) uses one worker containing the complete CFA as *code block*.
- Distributing the verification decreases the needed time.
- Distributing the work helps finding 430 more proofs and 130 more alarms.
4 Evaluation  

Increasing Resources

- DCPA(S1) runs on setup 1 (8 cores, 33 GB), DCPA(S2) runs on setup 2 (16 cores, 67 GB).
- Needed time and used memory are similar.
- On average, DCPA(S2) processes 20% more messages than DCPA(S1).
- Increasing number of messages causes time loss (nondeterministic).
4 Evaluation  DCPA vs. Predicate Analysis

- Predicate Analysis is faster and uses less memory
- Message grow larger since abstraction is deactivated, thus the memory usage increases
- The abstraction allows the predicate analysis to finish faster
QUESTIONS?
Thank you for your attention!
Appendix

Schema of an *Analysis Worker*
Appendix