Minimization of Visibly Pushdown Automata Using Partial Max-SAT

Matthias Heizmann, Christian Schilling, Daniel Tischner

University of Freiburg, Germany
Trace abstraction / Ultimate Automizer

annotated program $\mathcal{P}$

$\mathcal{A} := \text{CFA}^1(\mathcal{P})$

$\mathcal{L}(\mathcal{A}) = \emptyset$?

choose $w \in \mathcal{L}(\mathcal{A})$

$\mathcal{P}$ correct

$\mathcal{P}$ incorrect counterexample $w$

construct $\mathcal{A}_w$ of infeasible traces s.t. $w \in \mathcal{A}_w$

$\mathcal{A} := \mathcal{A} \setminus \mathcal{A}_w = \mathcal{A} \cap \overline{\mathcal{A}_w}$

$w$ feasible?

1 CFA = control flow automaton
Trace abstraction / **Ultimate Automizer**

- Automaton $\mathcal{A}$ grows **exponentially** in number of iterations unless we apply minimization.

---

1 CFA = control flow automaton
Visibly pushdown automata (VPA)

- Programs with procedures
  Traces also contain calls and returns

- VPA: restricted pushdown automata
  Read words with three types of symbols
    - **internal** – “no stack”
    - **call** – “push current state”
    - **return** – “pop”

- VPA inherit nice properties of finite automata
  - Boolean operations
  - Decidability
However, **no minimization**!
Minimization

- Minimization = reduction (number of states)
- **Merge** states (according to a congruence)
- Preserve the language
Minimization of finite automata

\[(a + b)^* a (a + b)\]

non-minimal DFA
Minimization of finite automata

\[(a + b)^* a (a + b)\]

non-minimal DFA

minimal DFA / merge-minimal NFA
Minimization of finite automata

\[(a + b)^* a (a + b)\]

non-minimal DFA

minimal DFA / merge-minimal NFA

minimal NFA
Minimization of \( V_{\text{PA}} \)

1. Observation: Return transitions can sometimes be ignored.

2. Observation: Ignoring return transitions can destroy transitivity.
Minimization of $\mathcal{VPA}$

1. Observation:
Return transitions can sometimes be ignored

![Diagram showing a VPA with transitions labeled $c_1$, $c_2$, $r$, $a$, $q_0$, $q_1$, $q_2$, $q_3$, $q_f$.]
Minimization of $V_{PA}$

1. Observation:
Return transitions can sometimes be ignored

2. Observation:
Ignoring return transitions can destroy transitivity
3. Observation:
Merging call predecessors changes the stack alphabet
Congruence for minimization

- Two states are **equivalent** if they
  - are **both accepting** or **both non-accepting**
  - reach **equivalent states** under the same symbol
    - and **equivalent stack symbols** (for returns)

```
\begin{align*}
\text{p} & \xrightarrow{a} \text{p}' \\
\text{q} & \xrightarrow{a} \text{q}'
\end{align*}
```

```
\begin{align*}
\text{p} & \xrightarrow{r \uparrow \hat{p}} \text{p}' \\
\text{q} & \xrightarrow{r \uparrow \hat{q}} \text{q}'
\end{align*}
```
Congruence for minimization

- Two states are **equivalent** if they
  - are **both accepting** or **both non-accepting**
  - **reach equivalent states** under the same symbol
    - and **equivalent stack symbols** (for returns)

- How to compute such a relation?
  - Encode existence as **Boolean formula**
  - Any satisfying assignment represents a congruence
Encoding

- Boolean variables $X_{\{p,q\}}$ for any two states $p, q$
  - $p$ and $q$ can be merged if $X_{\{p,q\}}$ is true

- Constraints enforce that the relation
  - is an **equivalence relation**
  - is compatible with **acceptance condition**
  - is a **congruence for transition relation**
Equivalence relation

- Reflexivity
  \[ X\{q,q\} \] (1)

- Symmetry
  encoded in variables

- Transitivity
  \[ X\{q_1,q_2\} \land X\{q_2,q_3\} \rightarrow X\{q_1,q_3\} \] (2)
Compatibility with acceptance condition

- Accepting state $p \in F$ must not be merged with non-accepting state $q \notin F$

$$\neg \chi_{\{p,q\}}$$ (3)
Congruence for transition relation

- States are only merged if their successors are merged
  - Internal and call transitions

\[ X_{\{p, q\}} \rightarrow X_{\{p', q'\}} \]  \hspace{1cm} (4.1)
Congruence for transition relation

- States are only merged if their successors are merged
  - Return transitions

\[ X_{\{p,q\}} \land X_{\{\hat{p},\hat{q}\}} \rightarrow X_{\{p',q'\}} \]  \hspace{1cm} (4.2)

- Only required for reachable \( \hat{p}, \hat{q} \)
Are we done yet?

- Assignment

\[ X\{q,q\} \mapsto \text{true} \quad X\{p,q\} \mapsto \text{false} \quad (p \neq q) \]

corresponds to original \( V_{PA} \) – so sad!
Partial maximum satisfiability (PMax-SAT)

- Clauses are either hard or soft
- Assignment must satisfy
  - all hard clauses
  - as many soft clauses as possible

Consider all clauses so far as hard clauses
Add soft clauses $X \{p, q\}$ (5)

Rationale: Merge as many states as possible

Solution corresponds to a local optimum
PMax-SAT encoding

- Partial maximum satisfiability (PMax-SAT)
  - Clauses are either hard or soft
  - Assignment must satisfy
    - all hard clauses
    - as many soft clauses as possible

- Consider all clauses so far as hard clauses

- Add soft clauses

\[ X_{\{p,q\}} \] (5)

Rationale: Merge as many states as possible

- Solution corresponds to a local optimum
Integration in **Ultimate Automizer**

- 165 programs from **SV-COMP 2016**
- Resource limit: 300 s / 4 GiB

<table>
<thead>
<tr>
<th>minimization used?</th>
<th># solved</th>
<th>$\emptyset$ time total</th>
<th>$\emptyset$ time min.</th>
<th>$\emptyset$ removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>66</td>
<td>16,085</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>yes</td>
<td>same 66</td>
<td>15,564</td>
<td>2,649</td>
<td>3,077</td>
</tr>
<tr>
<td></td>
<td>+ 12</td>
<td>101,985</td>
<td>61,384</td>
<td>8,472</td>
</tr>
</tbody>
</table>

*times given in ms*
Automata from Ultimate Automizer

![Scatter plot with data points indicating relative reduction in states against input state count. The plot shows a comparison between deterministic and nondeterministic VPA models with 596 data points.](chart.png)
Recap

- Algorithm for reducing $V_{PA}$ by merging states
- Reduction to synthesis of language-preserving congruence
- Reduction to solving a Boolean optimization problem