Visualizing SMT-Based Parallel Constraint Solving

Jelena Budakovic, Amedeo Zucchetti, Matteo Marescotti, Antti E. J. Hyvärinen, and Natasha Sharygina

Università della Svizzera italiana
Switzerland

SMT@CAV2017
Motivations

• SMT
  • Very expressive language
  • Problem’s intrinsic high complexity (SAT + decision procedures)

• SMT solvers:
  • Widely used for modelling
  • Highly optimized sequentially

• Parallel SMT:
  • Aims to solve more and faster
  • Parallel computing is difficult!
Verification Use Case

- SMT Solver
- Model Checker
- SMT Viewer
- Source code repository

(PDR)
Parallelize the work

- Portfolio:
  - Many search-independent processes

- Partitioning:
  - The problem is partitioned to several sub-problems such that:
    - problem is SAT: exists a sub-problem SAT
    - problem is UNSAT: all sub-problems are UNSAT

- Combination of the two above
Parallelization Tree Framework
SAT 2015, ATVA 2016, FMCAD 2017 (P3)
Parallelization Tree Framework
SAT 2015, ATVA 2016, FMCAD 2017 (P3)

Circle: original instance
3 solvers portfolio
Parallelization Tree Framework

SAT 2015, ATVA 2016, FMCAD 2017 (P3)

- **Circle**: original instance
  - 3 solvers portfolio

- **Diamond**: partitioning
  - Each diamond represents a way to partition the parent’s instance.
Parallelization Tree Framework

SAT 2015, ATVA 2016, FMCAD 2017 (P3)

Circle: original instance
3 solvers portfolio

Diamond: partitioning
Each diamond represents a way to partition the parent’s instance.

Circle: a partition of the parent’s instance
Each node could possibly be the root of a new tree.
Parallelization Tree Framework
SAT 2015, ATVA 2016, FMCAD 2017 (P3)

**Circle:** original instance
3 solvers portfolio

**Diamond:** partitioning
Each diamond represents a way to partition the parent’s instance.

**Circle:** a partition of the parent’s instance
Each node could possibly be the root of a new tree.
Parallelization Tree Framework

SAT 2015, ATVA 2016, FMCAD 2017 (P3)

Circle: original instance
3 solvers portfolio

Diamond: partitioning
Each diamond represents a way to partition the parent’s instance.

Circle: a partition of the parent’s instance
Each node could possibly be the root of a new tree.
Parallelization Tree Framework

SAT 2015, ATVA 2016, FMCAD 2017 (P3)

Circle: original instance
3 solvers portfolio

Diamond: partitioning
Each diamond represents a way to partition the parent’s instance.

Circle: a partition of the parent’s instance
Each node could possibly be the root of a new tree.
Parallelization Tree Framework
SAT 2015, ATVA 2016, FMCAD 2017 (P3)

**Circle:** original instance
3 solvers *portfolio*

**Diamond:** partitioning
Each diamond represents a way to partition the parent’s instance.

**Circle:** a partition of the parent’s instance
Each node could possibly be the root of a new tree.
Parallelization Tree Framework

SAT 2015, ATVA 2016, FMCAD 2017 (P3)

Circle: original instance
3 solvers portfolio

Diamond: partitioning
Each diamond represents a way to partition the parent’s instance.

Circle: a partition of the parent’s instance
Each node could possibly be the root of a new tree.
Parallelization Tree Framework

SAT 2015, ATVA 2016, FMCAD 2017 (P3)

Circle: original instance
3 solvers portfolio

Diamond: partitioning
Each diamond represents a way to partition the parent’s instance.

Circle: a partition of the parent’s instance
Each node could possibly be the root of a new tree.
SMT Portfolio & Partitioning

• Portfolio:
  • Solver random seed

• Partitioning:
  • Search space partitioning (SAT)
  • Jointly exhaustive assumptions (*disjunction is a tautology*)
PDR Portfolio & Partitioning

• Portfolio:
  • PDR strategy
  • Underlying SMT solver’s random seed

• Partitioning:
  • PDR partitioning
PDR

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \quad \text{Init} \implies \neg \text{Bad} \]
PDR

\[\langle \text{Init}(X), Tr(X, X'), \text{Bad}(X)\rangle \quad \text{Init} \implies \neg \text{Bad}\]
PDR

\[ \langle \text{Init}(X), Tr(X, X'), \text{Bad}(X) \rangle \]

\[ \text{Init} \implies \neg \text{Bad} \]
PDR

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \quad \text{Init} \implies \neg \text{Bad} \]
PDR

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \]

\[ \text{Init} \implies \neg \text{Bad} \]
PDR

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \]

\[ \text{Init} \implies \neg \text{Bad} \]
PDR

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \quad \text{Init} \implies \neg \text{Bad} \]
PDR

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \]

\[ \text{Init} \implies \neg \text{Bad} \]
PDR

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \quad \text{Init} \implies \neg \text{Bad} \]
PDR

\langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \quad \text{Init} \implies \neg \text{Bad}
PDR Partitioning
FMCAD 2017

\langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle
PDR Partitioning

FMCAD 2017

\(\langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle\)
PDR Partitioning

FMCAD 2017

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \]
PDR Partitioning
FMCAD 2017
\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \]
PDR Partitioning

FMCAD 2017

\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}(X) \rangle \equiv \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}_1(X) \rangle \]
\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}_2(X) \rangle \]
\[ \langle \text{Init}(X), \text{Tr}(X, X'), \text{Bad}_3(X) \rangle \]
SMTViewer API

- SQLite database of events happened during parallel solving:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Integer</td>
<td>Unique event id number.</td>
</tr>
<tr>
<td>ts</td>
<td>Integer</td>
<td>The Unix timestamp of the event.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Instance name.</td>
</tr>
<tr>
<td>node</td>
<td>JSONArray&lt;Integer&gt;</td>
<td>Path from the root to the node.</td>
</tr>
<tr>
<td>event</td>
<td>String</td>
<td>Type of the event.</td>
</tr>
<tr>
<td>solver</td>
<td>String</td>
<td>Solver identifier.</td>
</tr>
<tr>
<td>data</td>
<td>JSONObject&lt;String, String&gt;</td>
<td>Data associated with the event.</td>
</tr>
</tbody>
</table>
Analysis Demo
Architecture

- **SMTService**:  
  - SMT: OpenSMT2  
  - PDR: Z3 Spacer
Architecture

**SMTService:**
- SMT: OpenSMT2
- PDR: Z3 Spacer

**SMTViewer:**
- Real time mode
- Analysis mode
Live Demo
Future work

- SMT formula visualization
- SMT learned clauses analysis
- PDR frame lemmas analysis
- SMTS: support for more solvers, reproducibility
- Any suggestions?
Thank you

**SMTS** public repository:

$ git clone https://scm.ti-edu.ch/repo/git/smts.git