Implementing Tableaux Using Binary Decision Diagrams

Seminar
28 May 2013

Joshua Chia
u5024740@anu.edu.au

Supervisors: Rajeev Goré, Jimmy Thomson
Seminar Overview

- Motivations
- Tableaux Methods for Modal Logic
- Binary Decision Diagrams
- BDD-based Tableaux
- Performance Comparison
- Review & Further Work
Motivations

Symbolic Model Checking:

SAT Solvers:

QBF Solvers:
- Olivio, O., Emerson, A. (2011)

Intuitionistic Logic Provers:
Tableaux Methods for Modal Logic

- Proof procedure for determining satisfiability
- Formula set is satisfiable if there exists an open branch
- Use Negated Normal Form (NNF) to simplify rules
- Static rules:
  \[
  \frac{P; \neg P; X}{\times} \quad (\wedge) \quad \frac{P \wedge Q; X}{P; Q; X} \quad (\vee) \quad \frac{P \vee Q; X}{P; X | Q; X}
  \]
- Transitional rule:
  \[
  (\Diamond \, K) \quad \frac{\Diamond P; \Box Q; X}{P; Q} \quad \forall r \cdot \Box r \notin X
  \]
Example of Tableaux Proof Procedure

- Not necessary to explore all branches (∨) but implicit branching on (◊K)
- Can be viewed a directed acyclical graph
Additional Tableaux Rules

- Global Assumptions Transition rule:
  \[
  \frac{\Diamond P; \Box Q; X}{P; Q; \Gamma} \quad \forall r \cdot \Box r \notin X
  \]

- Need to check for loops:
  \[
  \frac{p_0; \Diamond p_0}{p_0; \Diamond p_0} \quad (\Diamond \Gamma) \\
  \frac{p_0; \Diamond p_0}{(\Diamond \Gamma)} \\
  \frac{}{(\Diamond \Gamma)} \\
  \Gamma = \{\Diamond p_0\}
  \]

- Additional rules for **S4**:
  \[
  (T) \quad \frac{X; \Box P}{X; \Box P; P} \\
  (S4) \quad \frac{\Diamond P; \Box Q; X}{P; \Box Q} \quad \forall r \cdot \Box r \notin X
  \]
Binary Decision Diagrams

- Bryant, R. (1986)
- From Binary Decision Tree to (RO)BDD: $x_1 \land x_2$

- Reduced: uniqueness, irredundancy
- Ordered: fixed variable ordering for canonicity
Binary Decision Diagrams

- Compressed representation of Boolean formula, functions, sets and relations
- Operations perform on compressed representation
- Caveat: Variable Ordering matters and is NP-hard

BDD-based Tableaux

BDDs don’t capture modality:

- Abstraction: treat modal formula as “atomic”:
  \[ \Box P \overset{\text{def}}{=} a_{\Box P} \quad \Diamond P \overset{\text{def}}{=} a_{\Diamond P} \]

- Extend usage of NNF to Primitive Normal Form (PNF)
  \[
  \text{atoms}(\Diamond P) = \text{atoms}(\neg \Box \neg P) \\
  \text{atoms}(\neg \Box \neg P) = \{ a_{\Box \neg P} \} \cup \text{atoms}(\neg P)
  \]

BDDs as the underlying data structure for Tableaux

- Instead of representing explicit formula sets, use BDDs to represent the set of satisfiable sets
BDD-based Tableaux

- Earlier example:
  \[ \varphi = \lozenge p_0 \land \lozenge p_1 \land (\square \neg p_0 \lor p_2) \]
  \[ \text{atoms}(\varphi) = \{ p_0, p_1, p_2, \square \neg p_0, \square \neg p_1 \} \]

- Search tree:

```

[\lozenge p_0 \land \lozenge p_1 \land (\square \neg p_0 \lor p_2)]

\{p_2; \neg \square \neg p_0; \neg \square \neg p_1\} \leftrightarrow \{p_2; \lozenge p_0; \lozenge p_1\}
```

- [p_0]
- [p_1]

\{p_0\} \quad \{p_1\}

\square \text{BDD of formula set}
\circ \text{Satisfiable set}

\rightarrow (\lor)
\rightarrow (\lozenge K)
Performance Comparison

- For CPC problems, BDD approach is competitive
- FaCT++ dominates for modal heavy problems
- Optimised BDD mixed results: cost vs. benefit
- BDDs are memory-hungry: k_lin, k_path, k_ph, k_t4
- Comparable start-up overheads

<table>
<thead>
<tr>
<th>LWB Benchmark</th>
<th>Naive</th>
<th>BDD</th>
<th>BDD*</th>
<th>FaCT++</th>
<th>Out of</th>
</tr>
</thead>
<tbody>
<tr>
<td>k_branch_n</td>
<td>4</td>
<td>8</td>
<td>13</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>k_branch_p</td>
<td>2</td>
<td>9</td>
<td>21</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>k_d4_n</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_d4_p</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_dum_n</td>
<td>20</td>
<td>15</td>
<td>15</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_dum_p</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_grz_n</td>
<td>21</td>
<td>20</td>
<td>18</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_grz_p</td>
<td>7</td>
<td>12</td>
<td>14</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_lin_n</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_lin_p</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_path_n</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_path_p</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_ph_n</td>
<td>3</td>
<td>10</td>
<td>11</td>
<td>14</td>
<td>21</td>
</tr>
<tr>
<td>k_ph_p</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>8</td>
<td>21</td>
</tr>
<tr>
<td>k_poly_n</td>
<td>16</td>
<td>21</td>
<td>20</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_poly_p</td>
<td>15</td>
<td>21</td>
<td>19</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_t4p_n</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>k_t4p_p</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td><strong>Sum of above</strong></td>
<td><strong>151</strong></td>
<td><strong>197</strong></td>
<td><strong>202</strong></td>
<td><strong>336</strong></td>
<td><strong>378</strong></td>
</tr>
</tbody>
</table>

Tested: Intel Core2 Duo @ 3.0GHz, 8GB RAM, 64-bit Ubuntu
Performance: k_branch_n
Performance: k_dum_p

![Graph showing performance comparison between Naive, BDD, BDD*, and FaCT++](image)
Review & Further Work

Review:

- Explored potential application of BDDs to Tableaux provers for modal logic $\mathbf{K}$, $\mathbf{S4}$
- Implemented Naive and BDD-based Tableaux provers (GNU C, GLib, BuDDy)

Further Work:

- Domain-specific optimisations (Tableaux, BDD)
- Implementation-specific (avoid additional branches)
- Comparison with other state of art Tableaux provers
Thank You

Binary Decision Diagrams:

“one of the really fundamental data structures to come out in the last twenty-five years”

– Donald Knuth, 2008
Appendix
Modal Logic

- Modes of truth: necessarily □, possibly ◊
- Kripke Semantics:
  \[ \langle W, R, V \rangle, w \in W, V : P \leftrightarrow 2^w \]
  \[ w \models \square P \text{ iff } \forall v \in W, v \models P \lor vRw \]
  \[ w \models \diamond P \text{ iff } \exists v \in W, v \models P \land vRw \]
- Modal equivalences:
  \[ \diamond P \iff \neg \square \neg P \quad \square P \iff \neg \diamond \neg P \]
- In addition to the basic modal logic \( \textbf{K} \), our scope also included extension \( \textbf{S4} \)
Demonstration of Additional Branches

- Traditional Tableaux vs. BDD-based Tableaux:

\[
\frac{\Box p \lor q}{\Box p \mid q} \quad (\lor)
\]

\[
\{\Box p\} \quad \{\neg \Box p; q\} \Leftrightarrow \{\Diamond \neg p; q\}
\]

\[
[\neg p]
\]

\[
\{\neg p\}
\]

- Still correct, but unnecessary
Time Complexity of BDD Operations


\[
\begin{align*}
\text{mk}(i, u_0, u_1) & : O(1) \\
\text{build}(t) & : O(2^n) \\
\text{apply}(op, u_0, u_1) & : O(|u_1| \times |u_2|) \\
\text{restrict}(u, j, b) & : O(|u|) \quad \text{*} \\
\text{satCount}(u) & : O(|u|) \quad \text{*} \\
\text{anySat}(u) & : O(|p|) \\
\text{allSat}(u) & : O(|r| \times n) \\
\text{simplify}(d, u) & : O(|d| \times |u|) \quad \text{*}
\end{align*}
\]

* If dynamic programming is used