On the Role of Canonicity in Knowledge Compilation

Guy Van den Broeck and Adnan Darwiche

Jan 28, 2015, AAAI
Knowledge Compilation

• Reasoning with logical knowledge bases
• Tractable languages and compilers
• Boolean circuits: OBDDs, d-DNNFs, SDDs, etc.
• Applications:
  – Diagnosis
  – Planning
  – Inference in probabilistic databases, graphical models, probabilistic programs
  – Learning tractable probabilistic models
Bottom-Up Compilation with Apply

• Build Boolean **combinations** of existing circuits
• Compile CNF: (1) circuit for literals (2) disjoin to get circuit for clauses (3) conjoin for CNF.
• Compile arbitrary sentence incrementally

\[
(A \oplus (B \land D)) \land (C \lor D) = (A \oplus (B \land D)) \land (C \lor D)
\]

• Avoiding CNF crucial for many applications
Two Properties Under Investigation

Polytime Apply

Complexity is polynomial in size of input circuits.
Informally: one Apply cannot blow up size.

\[ \begin{array}{c}
\wedge \\
= O( \begin{array}{c}
\wedge \\
\times \\
\end{array} )
\end{array} \]

Canonicity

Equivalent sentences have identical circuits.

\[ A \wedge (C \lor D) \equiv (A \wedge C) \lor (A \wedge D) \]
What We Knew Before

• A practical language for bottom-up compilation requires a polytime Apply.
  – Explains success of OBDDs
  – Why do Apply when it blows up?
  – Guided search for new languages (structured DNNF)

• Canonicity is convenient for building compilers
  – Detect/cache equivalent subcircuits
What We Knew Before

- A practical language for bottom-up compilation requires a polytime Apply.
  - Explains success of OBDDs
  - Why do Apply when it blows up?
  - Guided search for new languages (structured DNNF)

- Canonicity is convenient for building compilers
  - Detect/cache equivalent subcircuits
Sentential Decision Diagrams

Properties:
- OBDD $\subseteq$ SDD
- Treewidth upper bound
- Quasipolynomial separation with OBDD
- Supports OBDD queries
Sentential Decision Diagrams

\[ f(A, B, C, D) = (A \oplus (B \land D)) \land C \]
Sentential Decision Diagrams

\[ f(A, B, C, D) = (A \oplus (B \land D)) \land C \]
Basing Decisions on Sentences

\[ f(A, B, C, D) = (A \land B) \lor (C \land D) \]

\[ A = t, \quad B = f, \quad C = t, \quad D = t \]
Basing Decisions on Sentences

\[ f(A, B, C, D) = (A \land B) \lor (C \land D) \]

\( A = t, B = f, C = t, D = t \)
Basing Decisions on Sentences

\[ f(A, B, C, D) = (A \land B) \lor (C \land D) \]

\[ A = t, \ B = f, \ C = t, \ D = t \]

Diagram of logical decisions based on the given conditions.
Basing Decisions on Sentences

\[ f(A, B, C, D) = (A \land B) \lor (C \land D) \]

A = t, B = f, C = t, D = t
Basing Decisions on Sentences

\[ f(A, B, C, D) = (A \land B) \lor (C \land D) \]

In an \((X, Y)\)-partition:

\[ f(X, Y) = p_1(X) s_1(Y) \lor \ldots \lor p_n(X) s_n(Y) \]

primes are **mutually exclusive, exhaustive** and not false
Compression and Canonicity

- An \((X,Y)\)-partition:
  \[
  f(X, Y) = p_1(X)s_1(Y) \lor \ldots \lor p_n(X)s_n(Y)
  \]
  is \textit{compressed} when the subs are distinct:
  \[
  s_i(Y) \neq s_j(Y) \text{ if } i \neq j
  \]
  - \(f(X,Y)\) has a \textit{unique} compressed \((X,Y)\)-partition
  - For fixed \(X,Y\) throughout the SDD (i.e. a vtree), compressed SDDs* are \textit{canonical}!

* requires some additional maintenance (pruning/normalization)
Compression

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]

\[ X = \{A, B\}, \quad Y = \{C, D\} \]
Compression

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]

\[ X = \{ A, B \}, \quad Y = \{ C, D \} \]

<table>
<thead>
<tr>
<th>prime</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \land B )</td>
<td>( A \land B )</td>
</tr>
<tr>
<td>( A \land \overline{B} )</td>
<td>( A \land \overline{B} )</td>
</tr>
<tr>
<td>( \overline{A} \land B )</td>
<td>( \overline{A} \land B )</td>
</tr>
<tr>
<td>( \overline{A} \land \overline{B} )</td>
<td>( \overline{A} \land \overline{B} )</td>
</tr>
<tr>
<td>( A \land B )</td>
<td>( A \land B )</td>
</tr>
</tbody>
</table>
Compression

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]

\[ X = \{ A, B \}, \quad Y = \{ C, D \} \]

<table>
<thead>
<tr>
<th>prime</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \land B )</td>
<td>true</td>
</tr>
<tr>
<td>( A \land \overline{B} )</td>
<td>( C \land D )</td>
</tr>
<tr>
<td>( \overline{A} \land B )</td>
<td>( C )</td>
</tr>
<tr>
<td>( \overline{A} \land \overline{B} )</td>
<td>( C \land D )</td>
</tr>
</tbody>
</table>
Compression

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]

\[ X = \{A, B\}, \quad Y = \{C, D\} \]

<table>
<thead>
<tr>
<th>prime</th>
<th>sub</th>
<th>prime</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A \land B)</td>
<td>true</td>
<td>(A \land B)</td>
<td>true</td>
</tr>
<tr>
<td>(A \land \overline{B})</td>
<td>(C \land D)</td>
<td>(\overline{A} \land B)</td>
<td>(C)</td>
</tr>
<tr>
<td>(\overline{A} \land B)</td>
<td>(C)</td>
<td>(\overline{B})</td>
<td>(C \land D)</td>
</tr>
</tbody>
</table>
Compression

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]

\[ X = \{A, B\}, \quad Y = \{C, D\} \]

<table>
<thead>
<tr>
<th>prime</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A \land B)</td>
<td>true</td>
</tr>
<tr>
<td>(A \land \overline{B})</td>
<td>(C \land D)</td>
</tr>
<tr>
<td>(\overline{A} \land B)</td>
<td>(C)</td>
</tr>
<tr>
<td>(\overline{A} \land \overline{B})</td>
<td>(C \land D)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>prime</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A \land B)</td>
<td>true</td>
</tr>
<tr>
<td>(\overline{A} \land B)</td>
<td>(C)</td>
</tr>
<tr>
<td>(\overline{B})</td>
<td>(C \land D)</td>
</tr>
</tbody>
</table>
Is Apply for SDDs Polytime?

**Algorithm 1 Apply(α, β, ◦)**

1:  if α and β are constants or literals then
2:    return α ◦ β     // result is a constant or literal
3:  else if Cache(α, β, ◦) ≠ nil then
4:    return Cache(α, β, ◦)     // has been computed before
5:  else
6:    γ ← {}  
7:    for all elements (p_i, s_i) in α do  
8:      for all elements (q_j, r_j) in β do  
9:        p ← Apply(p_i, q_j, ◦)         
10:       if p is consistent then  
11:         s ← Apply(s_i, r_j, ◦)    
12:         add element (p, s) to γ  
13:     // get unique decision node and return it
14:    return Cache(α, β, ◦) ← UniqueD(γ)
Is Apply for SDDs Polytime?

- $|\alpha|\times|\beta|$ recursive calls
- Polytime!
Is Apply for SDDs Polytime?

|α|𝑥|β| recursive calls
• Polytime!
• But what about compression/canonicity?

**Algorithm 1 Apply(α, β, ⋄)**

1: if α and β are constants or literals then
2: return α ⋄ β // result is a constant or literal
3: else if Cache(α, β, ⋄) ≠ nil then
4: return Cache(α, β, ⋄) // has been computed before
5: else
6: γ ← {}
7: for all elements (pᵢ, sᵢ) in α do
8: for all elements (qⱼ, rⱼ) in β do
9: p ← Apply(pᵢ, qⱼ, ∧)
10: if p is consistent then
11: s ← Apply(sᵢ, rⱼ, ⋄)
12: add element (p, s) to γ
13: // get unique decision node and return it
14: return Cache(α, β, ⋄) ← UniqueD(γ)
Is Apply for SDDs Polytime?

- Polytime Apply?
- Open question answered in this paper

**Algorithm 1** \texttt{Apply}(\(\alpha, \beta, \circ\))

1: if \(\alpha\) and \(\beta\) are constants or literals then
2: \hspace{1em} return \(\alpha \circ \beta\) // result is a constant or literal
3: else if \(\text{Cache}(\alpha, \beta, \circ) \neq \text{nil}\) then
4: \hspace{1em} return \(\text{Cache}(\alpha, \beta, \circ)\) // has been computed before
5: else
6: \hspace{1em} \(\gamma \leftarrow \{\}\)
7: \hspace{2em} for all elements \((p_i, s_i)\) in \(\alpha\) do
8: \hspace{3em} for all elements \((q_j, r_j)\) in \(\beta\) do
9: \hspace{4em} \(p \leftarrow \text{Apply}(p_i, q_j, \land)\)
10: \hspace{3em} if \(p\) is consistent then
11: \hspace{4em} \(s \leftarrow \text{Apply}(s_i, r_j, \circ)\)
12: \hspace{3em} add element \((p, s)\) to \(\gamma\)
13: \hspace{1em} (optionally) \(\gamma \leftarrow \text{Compress}(\gamma)\) // compression
\hspace{4em} // get unique decision node and return it
14: return \(\text{Cache}(\alpha, \beta, \circ) \leftarrow \text{UniqueD}(\gamma)\)
Theoretical Results

Theorem:
The class of Boolean functions $f_m(X_1, \ldots, X_m)$ such that $f_m$ has an SDD of size $O(m^2)$, yet the canonical SDD of $f_m$ has size $\Omega(2^m)$. 

<table>
<thead>
<tr>
<th>Notation</th>
<th>Transformation</th>
<th>SDD</th>
<th>Canonical SDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>conditioning</td>
<td>√</td>
<td>●</td>
</tr>
<tr>
<td>FO</td>
<td>forgetting</td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>SFO</td>
<td>singleton forgetting</td>
<td>√</td>
<td>●</td>
</tr>
<tr>
<td>∧C</td>
<td>conjunction</td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>∧BC</td>
<td>bounded conjunction</td>
<td>√</td>
<td>●</td>
</tr>
<tr>
<td>∨C</td>
<td>disjunction</td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>∨BC</td>
<td>bounded disjunction</td>
<td>√</td>
<td>●</td>
</tr>
<tr>
<td>¬C</td>
<td>negation</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
Two options

1. Enable compression
   – No polytime Apply
   – Canonicity

2. Disable compression
   – Polytime Apply
   – No Canonicity

What should we do? Popular belief:
Choose polytime Apply, or circuits blow up!
## Empirical Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Variables</th>
<th>Clauses</th>
<th>Compressed SDDs</th>
<th>SDD Size Compressed SDDs</th>
<th>Uncompressed SDDs</th>
<th>Compilation Time Compressed SDDs</th>
<th>Uncompressed SDDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C17</td>
<td>17</td>
<td>30</td>
<td>99</td>
<td>171</td>
<td>286</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>majority</td>
<td>14</td>
<td>35</td>
<td>123</td>
<td>193</td>
<td>384</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>b1</td>
<td>21</td>
<td>50</td>
<td>166</td>
<td>250</td>
<td>514</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>cm152a</td>
<td>20</td>
<td>49</td>
<td>149</td>
<td>3,139</td>
<td>18,400</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>cm82a</td>
<td>25</td>
<td>62</td>
<td>225</td>
<td>363</td>
<td>683</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>cm151a</td>
<td>44</td>
<td>100</td>
<td>614</td>
<td>1,319</td>
<td>24,360</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>cm42a</td>
<td>48</td>
<td>110</td>
<td>394</td>
<td>823</td>
<td>276,437</td>
<td>0.03</td>
<td>0.10</td>
</tr>
<tr>
<td>cm138a</td>
<td>50</td>
<td>114</td>
<td>463</td>
<td>890</td>
<td>9,201,336</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>decod</td>
<td>41</td>
<td>122</td>
<td>471</td>
<td>810</td>
<td>1,212,302</td>
<td>0.04</td>
<td>1.40</td>
</tr>
<tr>
<td>tcon</td>
<td>65</td>
<td>136</td>
<td>596</td>
<td>1,327</td>
<td>618,947</td>
<td>0.05</td>
<td>0.33</td>
</tr>
<tr>
<td>parity</td>
<td>61</td>
<td>135</td>
<td>549</td>
<td>978</td>
<td>2,793</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>cmb</td>
<td>62</td>
<td>147</td>
<td>980</td>
<td>2,311</td>
<td>81,980</td>
<td>0.12</td>
<td>0.02</td>
</tr>
<tr>
<td>cm163a</td>
<td>68</td>
<td>157</td>
<td>886</td>
<td>1,793</td>
<td>21,202</td>
<td>0.06</td>
<td>0.00</td>
</tr>
<tr>
<td>pcle</td>
<td>66</td>
<td>156</td>
<td>785</td>
<td>1,757</td>
<td>12,150,626</td>
<td>0.08</td>
<td>19.87</td>
</tr>
<tr>
<td>x2</td>
<td>62</td>
<td>166</td>
<td></td>
<td></td>
<td></td>
<td>0.08</td>
<td>19.87</td>
</tr>
<tr>
<td>cm85a</td>
<td>77</td>
<td>176</td>
<td>1,015</td>
<td>2,098</td>
<td>19,057</td>
<td>0.08</td>
<td>33.32</td>
</tr>
<tr>
<td>cm162a</td>
<td>73</td>
<td>173</td>
<td>907</td>
<td>2,050</td>
<td>153,228</td>
<td>0.08</td>
<td>0.16</td>
</tr>
<tr>
<td>cm150a</td>
<td>84</td>
<td>202</td>
<td>1,603</td>
<td>5,805</td>
<td>17,265,164</td>
<td>0.16</td>
<td>60.37</td>
</tr>
<tr>
<td>pcle8</td>
<td>98</td>
<td>220</td>
<td>1,518</td>
<td>4,335</td>
<td>15,532,667</td>
<td>0.18</td>
<td>33.32</td>
</tr>
<tr>
<td>eu</td>
<td>94</td>
<td>235</td>
<td>1,466</td>
<td>5,789</td>
<td>n/a</td>
<td>0.19</td>
<td>n/a</td>
</tr>
<tr>
<td>pm1</td>
<td>105</td>
<td>245</td>
<td>1,810</td>
<td>3,699</td>
<td>n/a</td>
<td>0.27</td>
<td>n/a</td>
</tr>
<tr>
<td>mux</td>
<td>73</td>
<td>240</td>
<td>1,825</td>
<td>6,517</td>
<td>n/a</td>
<td>0.19</td>
<td>n/a</td>
</tr>
<tr>
<td>cc</td>
<td>115</td>
<td>265</td>
<td>1,451</td>
<td>6,938</td>
<td>n/a</td>
<td>0.22</td>
<td>n/a</td>
</tr>
<tr>
<td>unreg</td>
<td>149</td>
<td>336</td>
<td>3,056</td>
<td>668,531</td>
<td>n/a</td>
<td>0.66</td>
<td>263.06</td>
</tr>
<tr>
<td>ldd</td>
<td>145</td>
<td>414</td>
<td>1,610</td>
<td>2,349</td>
<td>n/a</td>
<td>0.23</td>
<td>n/a</td>
</tr>
<tr>
<td>count</td>
<td>185</td>
<td>425</td>
<td>4,168</td>
<td>51,639</td>
<td>n/a</td>
<td>1.05</td>
<td>n/a</td>
</tr>
<tr>
<td>comp</td>
<td>197</td>
<td>475</td>
<td>2,212</td>
<td>4,500</td>
<td>205,105</td>
<td>0.24</td>
<td>0.22</td>
</tr>
<tr>
<td>f51m</td>
<td>108</td>
<td>511</td>
<td>3,290</td>
<td>6,049</td>
<td>n/a</td>
<td>0.52</td>
<td>0.32</td>
</tr>
<tr>
<td>myadder</td>
<td>212</td>
<td>612</td>
<td>2,793</td>
<td>4,408</td>
<td>35,754</td>
<td>0.24</td>
<td>0.04</td>
</tr>
<tr>
<td>cht</td>
<td>205</td>
<td>650</td>
<td>4,832</td>
<td>13,311</td>
<td>n/a</td>
<td>1.24</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Empirical Results

(a) Compressed SDDs

(b) Uncompressed SDDs

(a) Compressed SDDs

(b) Uncompressed SDDs
What We Know Now

• Canonical SDDs have **no polytime Apply!**
• Yet they work!
  Outperform OBDDs and non-canonical SDDs
• **We argue: Canonicity** is more important
  Facilitates caching and minimization (vtree search)
• Questions common wisdom
Thanks