Sentential Decision Diagrams and their Applications

Guy Van den Broeck, Arthur Choi, and Adnan Darwiche

UCLA

Nov 4, 2015, INFORMS
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

\[ p_1 s_1 \quad p_2 s_2 \quad p_3 s_3 > 50 \text{ Rep.} \]
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

> 50  Veto
Rep.
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

- If > 50 Rep., Veto
- If 48-50 Rep., Convince Indeps.
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

- p1 s1: > 50 Rep. Veto
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

- **p1** s1: > 50 Rep. Veto
- **p3** s3: < 48 Rep. Vote Nay
Basing Decisions on Sentences

- p1, s1
- p2, s2
- p3, s3
Basing Decisions on Sentences

Branch on sentences p1, p2, and p3:
Basing Decisions on Sentences

Branch on sentences \( p_1, p_2, \) and \( p_3 \):

- \( p_1, p_2, p_3 \) are **mutually exclusive, exhaustive** and not false
Basing Decisions on Sentences

Branch on sentences p1, p2, and p3:

- p1, p2, p3 are **mutually exclusive, exhaustive** and not false
- p1, p2, p3 are called **primes** and represented by SDDs
Basing Decisions on Sentences

Branch on sentences $p_1$, $p_2$, and $p_3$:

- $p_1$, $p_2$, $p_3$ are **mutually exclusive, exhaustive** and not false
- $p_1$, $p_2$, $p_3$ are called **primes** and represented by SDDs
- $s_1$, $s_2$, $s_3$ are called **subs** and represented by SDDs
Basing Decisions on Sentences

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

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

\[ f(A, B, C, D) = (A \land B) \lor (C \land D) \]
SDDs as Boolean Circuits

\[ f(A, B, C, D) = (A \oplus (B \land D)) \land C \]
(X,Y)-Partitions

US Senate: 54 Rep., 44 Dem., and 2 Indep.

- p1, s1: > 50 Rep. Veto
- p3, s3: < 48 Rep. Vote Nay
(X,Y)-Partitions

US Senate: 54 Rep., 44 Dem., and 2 Indep.

- **p1 s1**: > 50 Rep. → Veto
- **p3 s3**: < 48 Rep. → Vote Nay
(X,Y)-Partitions

US Senate: 54 Rep., 44 Dem., and 2 Indep.

- p1(X) s1(Y) > 50 Rep. Veto
- p3(X) s3(Y) < 48 Rep. Vote Nay
\[(X, Y)\text{-Partitions}

US Senate: 54 Rep., 44 Dem., and 2 Indep.

\[f (X, Y) = p_1(X) s_1(Y) \lor \ldots \lor p_n(X) s_n(Y)\]
Variable \textbf{order} becomes variable \textbf{tree} (vtree)

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]
Variable **order** becomes variable **tree** (vtree)

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]
Variable **order** becomes variable **tree** (vtree)

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]
Variable **order** becomes variable **tree** (vtree)

\[ f = (A \land B) \lor (B \land C) \lor (C \land D) \]
Variable **order** becomes variable **tree** (vtree)

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

right-linear vtree
OBDDs are SDDs

right-linear vtree
OBDDs are SDDs

right-linear vtree
Ingredients for Delicious Decision Diagrams

- Minimization
- Apply Function
- Succinctness
- Queries
Ingredients for Delicious Decision Diagrams

- Minimization
- Apply Function
- Succinctness
- Queries
Compression

• 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 subs are distinct: \(s_i(Y) \neq s_j(Y)\) if \(i \neq j\)

• \(f(X,Y)\) has a \textbf{unique} compressed \((X,Y)\)-partition
Compression

- An \((X,Y)\)-partition: \(f(X, Y) = p_1(X)s_1(Y) \lor \ldots \lor p_n(X)s_n(Y)\)

  is **compressed** when subs are distinct: \(s_i(Y) \neq s_j(Y)\) if \(i \neq j\)

- \(f(X,Y)\) has a **unique** compressed \((X,Y)\)-partition

<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

• An \((X, Y)\)-partition: \(f (X, Y) = p_1(X)s_1(Y) \lor \ldots \lor p_n(X)s_n(Y)\) is **compressed** when subs are distinct: \(s_i(Y) \neq s_j(Y)\) if \(i \neq j\)

• \(f(X, Y)\) has a **unique** compressed \((X, Y)\)-partition

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

For a fixed vtree (fixing $X,Y$ throughout the SDD), compressed SDDs are canonical!

Equivalent sentences have identical circuits.

$A \land (C \lor D) \equiv (A \land C) \lor (A \land D)$

$\quad\quad\quad = \quad\quad\quad$
OBDD Minimization

- 24 **ordering** of 4 variables

  - ABCD \(\Rightarrow\) ABDC \(\Rightarrow\) ADBC \(\Rightarrow\) DABC \(\Rightarrow\) DACB \(\Rightarrow\) ADCB
  - ACDB \(\Rightarrow\) ACBD \(\Rightarrow\) CABD \(\Rightarrow\) CADB \(\Rightarrow\) CDAB \(\Rightarrow\) DCAB
  - DCBA \(\Rightarrow\) CDBA \(\Rightarrow\) CBDA \(\Rightarrow\) CBAD \(\Rightarrow\) BCAD \(\Rightarrow\) BCDA
  - BDCA \(\Rightarrow\) DBCA \(\Rightarrow\) DBAC \(\Rightarrow\) BDAC \(\Rightarrow\) BADC \(\Rightarrow\) BACD

- 24 OBDDs for every function over 4 variables

- Searching for an optimal OBDD is searching for an **optimal variable order**
SDD Minimization

Diagram showing the process of SDD Minimization, including steps such as rotate, swap, and lrotate.
Ingredients for Delicious Decision Diagrams

- Minimization
- Apply Function
- Succinctness
- Queries
Efficient Apply Function

• Build Boolean **combinations** of existing circuits

• Compile **arbitrary** sentence incrementally

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

• **Polytime** Apply: one Apply cannot blow up size

\[
\text{size}^\land \text{size} = O(\text{size} \times \text{size})
\]
Is Apply for SDDs Polytime?

• $|\alpha| \times |\beta|$ recursive calls
• Polytime!

**Algorithm 1** Apply($\alpha$, $\beta$, $\circ$)

1: if $\alpha$ and $\beta$ are constants or literals then
2:   return $\alpha \circ \beta$  // result is a constant or literal
3: else if Cache($\alpha$, $\beta$, $\circ$) \neq \text{nil} then
4:   return Cache($\alpha$, $\beta$, $\circ$)  // has been computed before
5: else
6:   $\gamma \leftarrow \{\}$
7:   for all elements $(p_i, s_i)$ in $\alpha$ do
8:     for all elements $(q_j, r_j)$ in $\beta$ do
9:       $p \leftarrow \text{Apply}(p_i, q_j, \land)$
10:      if $p$ is consistent then
11:        $s \leftarrow \text{Apply}(s_i, r_j, \circ)$
12:        add element $(p, s)$ to $\gamma$
13:  return Cache($\alpha$, $\beta$, $\circ$) $\leftarrow$ UniqueD($\gamma$)
Ingredients for Delicious Decision Diagrams

- Minimization
- Apply Function
- Succinctness
- Queries
Succinctness

• Theory
  – OBDD $\subset$ SDD thus SDD never larger than OBDD
  – Quasi-polynomial separation with OBDD
    \textit{OBDD can be much larger than SDD}
  – Treewidth upper bounds (important in AI!)

• Practice
  – SDD Compiler available and effective
  – SDD Package: \url{http://reasoning.cs.ucla.edu/sdd/}
  – Can obtain orders of magnitude improvements
Ingredients for Delicious Decision Diagrams

- Minimization
- Apply Function
- Succinctness
- Queries
Queries

• OBDDs are Swiss army knife of supported queries
• SDDs are **equally powerful**

<table>
<thead>
<tr>
<th>Query</th>
<th>Description</th>
<th>OBDD</th>
<th>SDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>consistency</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VA</td>
<td>validity</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CE</td>
<td>clausal entailment</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IM</td>
<td>implicant check</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>EQ</td>
<td>equivalence check</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CT</td>
<td>model counting</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SE</td>
<td>sentential entailment</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ME</td>
<td>model enumeration</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

• Some enabled by canonicity + apply
• E.g., (Weighted) Model Counting for Probabilistic reasoning (E.g., $Pr(bill\ passes|Vote1=Yea)$)
Application: Bayesian Networks

• Incrementally compile network

![Bayesian Network Diagram]

- Visit To Asia
- Tuberculosis
- Tuberculosis or Cancer
- XRay Result
- Smoking
- Lung Cancer
- Bronchitis
- Dyspnea
Application: Bayesian Networks

- Incrementally compile network
Application: Bayesian Networks

• Incrementally compile network
Application: Bayesian Networks

- Incrementally compile network

![Diagram of Bayesian Network]

\[ M \land A = \]
Application: Bayesian Networks

- Incrementally compile network $M \wedge A$
Application: Bayesian Networks

• Incrementally compile network
• Compute probability of any query
Application: Bayesian Networks

- Incrementally compile network
- Compute probability of any query
- Better than state of the art (treewidth)
Application: Probabilistic Programming

Model = program with random numbers

reach(X,Y) :- flight(X,Y).
reach(X,Y) :- flight(X,Z), reach(Z,Y).

State of the art inference: SDDs
Application: Tractable Learning

• Given: data

• Objective:
  – learn a probability distribution
  – ensure distribution is tractable for querying

• Unstructured space: Voting data

• Structured space: Movie recommendation
Learning in Unstructured Spaces

• Voting data from US House
  1764 votes of 453 congressmen
• Learn distribution (Markov network)
• Represent as SDD to ensure tractability
• Query efficiency

![Graph showing timeout vs. % Unknown votes]
Learning in Structured Spaces

Student enrollment constraints:

• Must take at least one of Probability or Logic.

\[ P \lor L \]

• Probability is a prerequisite for AI.

\[ A \Rightarrow P \]

• The prerequisites for KR is either AI or Logic.

\[ K \Rightarrow (P \lor L) \]

\[ w = A \land K \land L \land \neg P \quad \text{impossible} \]

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>K</th>
<th>P</th>
<th>A</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>54</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1: Student enrollment data.
<table>
<thead>
<tr>
<th>rank</th>
<th>user 1</th>
<th>rank</th>
<th>user 2</th>
<th>rank</th>
<th>user 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Godfather</td>
<td>1</td>
<td>Star Wars V: The Empire Strikes Back</td>
<td>1</td>
<td>The Usual Suspects</td>
</tr>
<tr>
<td>2</td>
<td>Raiders of the Lost Ark</td>
<td>2</td>
<td>Star Wars IV: A New Hope</td>
<td>2</td>
<td>One Flew over the Cuckoo’s Nest</td>
</tr>
<tr>
<td>3</td>
<td>Casablanca</td>
<td>3</td>
<td>The Godfather</td>
<td>3</td>
<td>The Godfather: Part II</td>
</tr>
<tr>
<td>4</td>
<td>The Shawshank Redemption</td>
<td>4</td>
<td>The Shawshank Redemption</td>
<td>4</td>
<td>Monty Python and the Holy Grail</td>
</tr>
<tr>
<td>5</td>
<td>Schindler’s List</td>
<td>5</td>
<td>The Usual Suspects</td>
<td>5</td>
<td>Star Wars IV: A New Hope</td>
</tr>
</tbody>
</table>

Learn rankings of movies (permutations):
Predict new movies given preferences
Distributions over Structured Spaces: PSDDs

- Domain Constraints
- SDD
- PSDD

parametrization
Distribution

```
<table>
<thead>
<tr>
<th>L</th>
<th>K</th>
<th>P</th>
<th>A</th>
<th>Pr(L, K, P, A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>6.00%</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>54.00%</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10.00%</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.40%</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1.00%</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.60%</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>17.6%</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4.00%</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2.40%</td>
</tr>
</tbody>
</table>
```
Distribution

<table>
<thead>
<tr>
<th>L</th>
<th>K</th>
<th>Pr(L, K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>20.00%</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>80.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L, K, P, A</th>
<th>Pr(L, K, P, A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 0, 0, 0</td>
<td>0.00%</td>
</tr>
<tr>
<td>0, 0, 1, 0</td>
<td>0.00%</td>
</tr>
<tr>
<td>0, 0, 1, 1</td>
<td>6.00%</td>
</tr>
<tr>
<td>0, 1, 0, 0</td>
<td>54.00%</td>
</tr>
<tr>
<td>0, 1, 0, 1</td>
<td>0.00%</td>
</tr>
<tr>
<td>0, 1, 1, 0</td>
<td>0.00%</td>
</tr>
<tr>
<td>0, 1, 1, 1</td>
<td>10.00%</td>
</tr>
<tr>
<td>1, 0, 0, 0</td>
<td>4.40%</td>
</tr>
<tr>
<td>1, 0, 0, 1</td>
<td>0.00%</td>
</tr>
<tr>
<td>1, 0, 1, 0</td>
<td>1.00%</td>
</tr>
<tr>
<td>1, 0, 1, 1</td>
<td>0.60%</td>
</tr>
<tr>
<td>1, 1, 0, 0</td>
<td>17.6%</td>
</tr>
<tr>
<td>1, 1, 0, 1</td>
<td>0.00%</td>
</tr>
<tr>
<td>1, 1, 1, 0</td>
<td>4.00%</td>
</tr>
<tr>
<td>1, 1, 1, 1</td>
<td>2.40%</td>
</tr>
</tbody>
</table>
Distribution

\[
\begin{array}{c|c|c}
L & K & \Pr(L, K) \\
\hline
0 & 0 & 0.00\% \\
0 & 1 & 0.00\% \\
1 & 0 & 20.00\% \\
1 & 1 & 80.00\% \\
\end{array}
\]

\[
\begin{array}{c|c|c}
P & A & \Pr(P, A) \\
\hline
0 & 0 & 73.33\% \\
0 & 1 & 0.00\% \\
1 & 0 & 16.67\% \\
1 & 1 & 10.00\% \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
L & K & P & A & \Pr(L, K, P, A) \\
\hline
0 & 0 & 0 & 0 & 0.00\% \\
0 & 0 & 1 & 0 & 0.00\% \\
0 & 1 & 0 & 0 & 6.00\% \\
0 & 0 & 1 & 1 & 54.00\% \\
0 & 1 & 0 & 1 & 0.00\% \\
0 & 1 & 1 & 0 & 0.00\% \\
0 & 1 & 1 & 1 & 10.00\% \\
1 & 0 & 0 & 0 & 4.40\% \\
1 & 0 & 1 & 0 & 1.00\% \\
1 & 0 & 1 & 1 & 0.60\% \\
1 & 1 & 0 & 0 & 17.6\% \\
1 & 1 & 0 & 1 & 0.00\% \\
1 & 1 & 1 & 0 & 4.00\% \\
1 & 1 & 1 & 1 & 2.40\%
\end{array}
\]
Reasoning with PSDDs
Example: Preference Distributions

observe:
• favorite movie is Star Wars V

<table>
<thead>
<tr>
<th>rank</th>
<th>movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Star Wars V: The Empire Strikes Back</td>
</tr>
<tr>
<td>2</td>
<td>Star Wars IV: A New Hope</td>
</tr>
<tr>
<td>3</td>
<td>The Godfather</td>
</tr>
<tr>
<td>4</td>
<td>The Shawshank Redemption</td>
</tr>
<tr>
<td>5</td>
<td>The Usual Suspects</td>
</tr>
</tbody>
</table>

observe:
• favorite movie is Star Wars V
• no other Star Wars movie in top-5
• at least one comedy in top-5

<table>
<thead>
<tr>
<th>rank</th>
<th>movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Star Wars V: The Empire Strikes Back</td>
</tr>
<tr>
<td>2</td>
<td>American Beauty</td>
</tr>
<tr>
<td>3</td>
<td>The Godfather</td>
</tr>
<tr>
<td>4</td>
<td>The Usual Suspects</td>
</tr>
<tr>
<td>5</td>
<td>The Shawshank Redemption</td>
</tr>
</tbody>
</table>
Conclusions

• SDD a strict **superset** of OBDD:
  – Characterized by trees, which include orders
  – Branch over sentences, which include literals

• SDDs maintain key **properties** of OBDDs:
  – Canonical, Polytime* Apply, Queries, etc.

• SDDs are more **succinct**
  – Treewidth instead of pathwidth

• Lots of **applications** in probabilistic AI and ML
References


• Xue, Yexiang, Arthur Choi, and Adnan Darwiche. "Basing decisions on sentences in decision diagrams." Twenty-Sixth AAAI Conference on Artificial Intelligence. 2012.


References

References


