 Knowledge Compilation

Jinbo Huang
NICTA and ANU
Is $A, \neg C$ a normal device behavior?
Propositional Reasoning

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C

Is A, ¬C a normal device behavior?

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C
A, ¬C, okX, okY

Satisfiability Algorithm

A. Darwiche
Satisfiability (SAT)

- **SAT Solvers:** Significant growth in last decade; many solvers publicly available (source code); millions of clauses not uncommon.

- **Applications:** Verification, planning, diagnosis, CAD, non-propositional reasoning (e.g., SMT), ...
Knowledge Compilation

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C

Compiler

Compiled Structure

Queries

Evaluator (Polytime)

A. Darwiche
Knowledge Compilation

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C
KnowledgeCompilation

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C

Compiler

PrimeImplicates
OBDD
...

Queries

Evaluator
(Polytime)

A. Darwiche
Knowledge Compilation Map

- What’s the space of possible target compilation languages?
  - Can it be synthesized in a semantically systematic way?

- How do the languages compare?
  - Succinctness (relative size)
  - Operations they support in polytime
Applications

- Diagnosis
  - Is this a normal behavior?
  - What are the possible faults?

- Planning
  - Can this goal be achieved?
  - Generate plan with highest reward
  - Generate plan with highest success probability

- Probabilistic reasoning
  - What is the probability of X given Y

- Formal verification / CAD:
  - Is it possible that the design will exhibit behavior X?
  - Are two designs equivalent?
For a given application: identify needed operations

Choose most succinct language that supports desired operations

Compile knowledge base into chosen language
Agenda

- Part I: Languages
- Part II: Operations
- Part III: Compilers
- Part IV: Applications
Part I: Languages
A Knowledge Compilation MAP

Negation Normal Form

Polytime Operations
- Consistency (CO)
- Validity (VA)
- Clausal entailment (CE)
- Sentential entailment (SE)
- Implicant testing (IP)
- Equivalence testing (EQ)
- Model Counting (CT)
- Model enumeration (ME)
- Projection (exist. quantification)
- Conditioning
- Conjoin, Disjoin, Negate

Succinctness
Propositional Logic

- Literal: \( X, \neg X \)
- Clause: \( (X \lor \neg Y \lor \neg Z) \)
- Term: \( (\neg X \land Y \land Z) \)
- **CNF**: Conjunctive Normal Form
  \( (X \lor \neg Y \lor \neg Z) \land \ldots \land (Y \lor \neg W) \)
- **DNF**: Disjunctive Normal Form
  \( (\neg X \land Y \land Z) \lor \ldots \lor (X \land \neg Z \land W) \)
Propositional Logic

- **Truth assignment (TA)**
  \[ X : \text{true} , Y : \text{false} , Z : \text{true} , W : \text{false} \]

- **TA satisfies sentence (model)**
  \[ (X \lor \neg Y \lor \neg Z) \land \ldots \land (Y \lor \neg W) \]

- **Following TA is not a model**
  \[ X : \text{true} , Y : \text{false} , Z : \text{true} , W : \text{true} \]
A Knowledge Compilation MAP

Negation Normal Form

- Decomposability
- Determinism
- Smoothness
- Flatness
- Decision
- Ordering

Polytime Operations

- Consistency (CO)
- Validity (VA)
- Clausal entailment (CE)
- Sentential entailment (SE)
- Implicant testing (IP)
- Equivalence testing (EQ)
- Model Counting (CT)
- Model enumeration (ME)

Projection (exist. quantification)
- Conditioning
- Conjoin, Disjoin, Negate

Succinctness

¬A ¬B A C ¬D D ¬C

and and and and or or or or or

A. Darwiche
Queries

- Consistency (CO)
- Validity (VA)
- Sentential entailment (SE)
- Clausal entailment (CE): KB implies clause
- Implicant testing (IP): term implies KB
- Equivalence testing (EQ)
- Model Counting (CT)
- Model enumeration (ME)
Transformations

- Projection (existential quantification)
- Conditioning
- Conjoin
- Disjoin
- Negate
Representation vs Compilation Languages

- Representation Language (intuitive):
  - CNF
  - DNF

- Target Compilation Language (tractable):
  - Binary Decision Diagrams (BDDs)
  - DNNF
Negation Normal Form

A. Darwiche
Negation Normal Form

Decomposability
Determinism
Smoothness
Flatness
Decision
Ordering

¬A  B  ¬B  A

c and and and and

c or or or or

C  ¬D  D  ¬C
Decomposability

A, B

¬¬ ¬¬

¬¬ ¬¬

B A

C

¬¬ ¬¬

¬¬ ¬¬

D D

¬¬ ¬¬

D ¬¬ C

Decomposability

A, Darwiche
NNF Subsets

NNF

CO, CE, ME

DNNF

A. Darwiche
Determinism

\[ \neg A \quad B \quad \neg B \quad A \]

\[ \text{and} \quad \text{and} \quad \text{and} \quad \text{and} \]

\[ \text{or} \quad \text{or} \quad \text{or} \quad \text{or} \]

\[ C \quad \neg D \quad D \quad \neg C \]
Smoothness

A. Darwiche
NNF Subsets

NNF

- d-NNF
- s-NNF

DNNF

- CO, CE, ME
- VA, IP, CT

d-DNNF

- EQ?

sd-DNNF
Nested vs Flat languages

\[(X \land Y \land Z) \lor (Z \lor \neg X \lor \neg Y) \lor (Y \land Z \lor \neg X) \lor (\neg X \lor Y \lor \neg Z)\]
Simple Conjunction

A. Darwiche

Simple conjunction implies decomposability
Simple Disjunction

A. Darwiche
NNF Subsets

- d-NNF
- s-NNF
- DNNF
- f-NNF

- CO, CE, ME
- VA, IP, CT
- d-DNNF
- sd-DNNF
- DNF
- CNF

A. Darwiche
Prime Implicates (PI)

Resolution

implication that:

\[
\frac{(\alpha \lor X), (\beta \lor \neg X)}{\alpha \lor \beta}
\]

- CNF:
  \[
  (\neg A \lor B) \land (\neg B \lor C) \land (\neg C \lor D)
  \]

- PI:
  \[
  (\neg A \lor B) \land (\neg B \lor C) \land (\neg C \lor D) \land
  (\neg A \lor C) \land (\neg A \lor D) \land (\neg B \lor D)
  \]
Prime Implicants (IP)

**Consensus**

\[
\frac{(\alpha \land X), (\beta \land \neg X)}{(\alpha \land \beta)}
\]

- **DNF:**
  \[
  (A \land B) \lor (\neg B \land C)
  \]

- **IP:**
  \[
  (A \land B) \lor (\neg B \land C) \lor (A \land C)
  \]
NNF Subsets

NNF

- d-NNF
- s-NNF
- DNNF
- f-NNF

- d-DNNF
- sd-DNNF
- DNF
- CNF

- CO, CE, ME
- VA, IP, CT

- EQ?
- VA, IP, SE, EQ

- CO, CE, ME
- VA, IP, SE, EQ

A. Darwiche
Decision

or

and

X

α

−X

β

α, β: Are decision nodes
Decision
Decision

\[
\text{or} \quad \text{and} \quad \text{and}
\]

\[
X \alpha \neg X \beta
\]

\[
X \quad \alpha \quad \beta
\]
Binary Decision Diagrams (BDDs)

Decision implies determinism
NNF Subsets

- d-NNF
- s-NNF
- DNNF
- f-NNF

- BDD
- d-DNNF
- sd-DNNF
- DNF
- CNF
- IP
- PI

- CO, CE, ME
- VA, IP, CT
- EQ?
- VA, IP, SE, EQ
- CO, CE, ME
- VA, IP, SE, EQ

A. Darwiche
Binary Decision Diagrams (BDDs)

\[
\begin{align*}
X_1 &\quad \neg X_1 \\
\neg X_2 &\quad \neg X_2 \\
\neg X_3 &\quad \neg X_3 \\
\end{align*}
\]

\[
\begin{align*}
X_2 &\quad X_2 \\
X_3 &\quad X_3 \\
true &\quad false \\
\end{align*}
\]

\[
\begin{align*}
\text{Decision + decomposability} &\quad = \text{FBDD} \\
\text{Test once property} &\quad \text{A. Darwiche}
\end{align*}
\]
NNF Subsets

- d-NNF
- s-NNF
- DNNF
- f-NNF

- BDD
- FBDD
- sd-DNNF
- MODS
- DNF
- IP
- CNF
- PI

- CO, CE, ME
- VA, IP, CT
- EQ?
- VA, IP, SE, EQ
- SE, EQ
- VA, IP, SE, EQ

A. Darwiche
Binary Decision Diagrams (BDDs)

Decision + decomposability + ordering = OBDD

A. Darwiche
NNF Subsets

- NNF
  - d-NNF
    - BDD
      - FBDD
        - OBDD
  - s-NNF
    - sd-DNNF
      - MODS
  - DNNF
  - f-NNF
    - CNF
      - PI

Attributes:
- CO, CE, ME
- VA, IP, CT
- EQ?
- SE, EQ
- VA, IP, SE, EQ
OBDD Example: Odd Parity

Symmetric Functions
Language Succinctness

L1 at least as succinct as L2

\[ L1 \leq L2 \]

Size \( p(n) \) \quad Size \( n \)

L1 is more succinct than L2

\[ L1 < L2 \]
Odd Parity Function

A. Darwiche
\( \text{sd-DNNF} \equiv \text{d-DNNF} \)

\( \text{DNNF} \rightarrow \text{CNF} \)

\( \text{DNF} \rightarrow \text{OBDD} \rightarrow \text{PI} \)

\( \text{OBDD} \leftrightarrow \text{DNF} \)

\( \text{PI} \leftrightarrow \text{DNNF} \)
Tractability & Succinctness

- NNF
  - DNNF: decomposability
  - d-DNNF: determinism
  - FBDD: decision
  - OBDD: ordering

Space Efficiency (succinctness)

Tractable Operations
- Diagnosis, Non-mon
- Probabilistic reasoning

A. Darwiche
Separating Functions

- **OBDD/FBDD:**
  - Hidden weighted bit function $hwb(x_1, \ldots, x_n)$

- **DNNF/DNF:**
  - Odd parity function $parity(x_1, \ldots, x_n)$

- **DNNF/OBDD:**
  - Distinct integers function $distinct(x_1, \ldots, x_n)$
Agenda

- Part I: Languages
- Part II: Operations
- Part III: Compilers
- Part IV: Applications
Part II: Operations
Knowledge Compilation

KB

Compiler

Compiled Structure

Queries

Evaluator (Polytime)

A. Darwiche
Queries

- Consistency (CO)
- Validity (VA)
- Sentential entailment (SE)
- Clausal entailment (CE): KB implies clause
- Implicant testing (IP): term implies KB
- Equivalence testing (EQ)
- Model Counting (CT)
- Model enumeration (ME)
Transformations

- Projection (existential quantification)
- Conditioning
- Conjoin
- Disjoin
- Negate
Decomposability
Decomposability

\begin{center}
\begin{tikzpicture}
  \node (root) {and};
  \node (left) [below left] {or}
    child {node (A) {and}
      child {node (A1) {$\neg A$}}
      child {node (B) {B}}
    }
    child {node (A2) {and}
      child {node (B1) {B}}
      child {node (A) {$\neg A$}}
    }
  
  \node (right) [below right] {or}
    child {node (C) {and}
      child {node (C1) {C}}
      child {node (D) {$\neg D$}}
    }
    child {node (D1) {and}
      child {node (D) {D}}
      child {node (C) {$\neg C$}}
    }
    child {node (E) {and}
      child {node (F) {and}}
      child {node (G) {and}}
    }
    child {node (H) {and}
      child {node (I) {and}}
      child {node (J) {and}}
    }
  
  \node (left2) [below] {or}
    child {node (A) {and}
      child {node (A1) {$\neg A$}}
      child {node (B) {B}}
    }
    child {node (A2) {and}
      child {node (B1) {B}}
      child {node (A) {$\neg A$}}
    }
  
  \node (right2) [below] {or}
    child {node (C) {and}
      child {node (C1) {C}}
      child {node (D) {$\neg D$}}
    }
    child {node (D1) {and}
      child {node (D) {D}}
      child {node (C) {$\neg C$}}
    }
    child {node (E) {and}
      child {node (F) {and}}
      child {node (G) {and}}
    }
    child {node (H) {and}
      child {node (I) {and}}
      child {node (J) {and}}
    }
  
  \node (left3) [below] {or}
    child {node (A) {and}
      child {node (A1) {$\neg A$}}
      child {node (B) {B}}
    }
    child {node (A2) {and}
      child {node (B1) {B}}
      child {node (A) {$\neg A$}}
    }
  
  \node (right3) [below] {or}
    child {node (C) {and}
      child {node (C1) {C}}
      child {node (D) {$\neg D$}}
    }
    child {node (D1) {and}
      child {node (D) {D}}
      child {node (C) {$\neg C$}}
    }
    child {node (E) {and}
      child {node (F) {and}}
      child {node (G) {and}}
    }
    child {node (H) {and}
      child {node (I) {and}}
      child {node (J) {and}}
    }
\end{tikzpicture}
\end{center}

A. Darwiche
Example Knowledge Base

\[
\begin{align*}
A & \land \text{okX} \rightarrow \neg B \\
\neg A & \land \text{okX} \rightarrow B \\
B & \land \text{okY} \rightarrow \neg C \\
\neg B & \land \text{okY} \rightarrow C
\end{align*}
\]
Decomposable
Decomposable

B
A, okX
C, okY
Decomposable
Satisfiability

- \( \text{SAT}(A \text{ or } B) \) iff \( \text{SAT}(A) \) or \( \text{SAT}(B) \)
- \( \text{SAT}(A \text{ and } B) \) iff \( \text{SAT}(A) \) and \( \text{SAT}(B) \)
- \( \text{SAT}(X) \) is true
- \( \text{SAT}(\neg X) \) is true
- \( \text{SAT}(\text{True}) \) is true
- \( \text{SAT}(\text{False}) \) is false
Satisfiability

\[
\begin{align*}
A &\lor A' \\
&\lor \neg B \\
&\lor \neg A' \\
&\lor \neg C \\
&\land \neg \text{okX} \\
&\land \neg \text{okY} \\
&\land \text{A} \\
&\land \text{C} \\
&\lor \text{B} \\
&\lor \text{C} \\
&\lor \text{okX} \\
&\lor \text{okY}
\end{align*}
\]
Satisfiability

A. Darwiche
Partial Decomposability

Decomposable except on \( \text{okZ} \)

\[
\begin{array}{c}
\text{or} \\
\text{and} \\
B \quad \text{or} \\
\sim A \\
\sim \text{okZ}
\end{array}
\quad
\begin{array}{c}
\text{or} \\
\text{and} \\
\sim \text{C} \\
A \\
\text{okZ}
\end{array}
\quad
\begin{array}{c}
\text{or} \\
\text{or} \\
\sim \text{B} \\
C \\
\text{okZ}
\end{array}
\quad
\begin{array}{c}
\text{or} \\
\text{and} \\
\sim \text{okZ}
\end{array}
\]
Clausal Entailment

\[ \text{KB} \text{ entails } L_1 \lor L_2 \lor \ldots \lor L_n \, ? \]

\[ \text{KB} \land \neg L_1 \land \neg L_2 \land \ldots \land \neg L_n \text{ SAT?} \]
Literal Conjoin

\[ (\neg A \lor \neg C \lor A \lor \neg B \lor \neg okX \lor okY) \]
Literal Conjoin

\[
\begin{align*}
\text{A} & \quad \text{or} \quad \text{A} \\
\text{and} & \quad \text{A} \\
\text{or} & \quad \text{A} \\
\text{and} & \quad \text{A} \\
\text{B} & \quad \text{or} \\
\text{or} & \quad \text{A} \\
\text{or} & \quad \text{A} \\
\text{or} & \quad \text{A} \\
\text{or} & \quad \text{A} \\
\text{~B} & \\
\text{~A} & \\
\text{~C} & \\
\text{~okX} & \\
\text{~okY} & \\
\text{C} & \\
\end{align*}
\]
Literal Conjoin

Conditioning

A

B or or or or or ~B

~C ~okX ~okY

false true
Literal Conjoin

```
and
   or
   and
   B or or or or
   or or or
   false ~C true
   ~okX ~okY
```

A. Darwiche
A \sim C \quad \text{ok}X \quad \text{ok}Y
Partial Decomposability

Decomposable except on okZ

Clausal entailment test works as long as clause mentioned all variables on which we don’t have decomposability!
(\text{okZ}) \land (B \lor (A \land (C \lor \neg B)))
\begin{center}
\begin{tikzpicture}
  \node {and}
  child {node {or}
    child {node {and}
      child {node {B}}
      child {node {or}
        child {node {\sim A}}
        child {node {true}}}
    }
    child {node {and}
      child {node {or}
        child {node {\sim C}}
        child {node {false}}}
      child {node {or}
        child {node {A}}
        child {node {false}}}
    }
    child {node {or}
      child {node {or}
        child {node {\sim B}}
      }
    }
  }
  child {node {~\text{okZ}}}
\end{tikzpicture}
\end{center}
Projection: Existential Quantification

Knowledge Base

\[ \Delta = A \rightarrow B, B \rightarrow C, C \rightarrow D \]

Existentially quantifying B,C

Forgetting B,C

Projecting on A,D

\[ (\exists B \exists C \Delta) = A \rightarrow D \]
Projection: Existential Quantification

Formal Definition

\[ \exists X \Delta = (\Delta \mid X) \lor (\Delta \mid \neg X) \]

- If Knowledge base is a CNF:
  - Close under resolution
  - Remove all clauses that mention X
Projection: Existential Quantification

A or A ~A ~C

B X Y C ~B

or

and

and

B or or or or ~B

~A ~C A C

~okX ~okY
Projection: Existential Quantification

\[ \exists X (\Delta \land \Gamma) \]

\[ = (\Delta \land \Gamma) \land X \lor (\Delta \land \Gamma) \land \neg X \]

\[ = (\Delta \land X) \land (\Gamma \land X) \lor (\Delta \land \neg X) \land (\Gamma \land \neg X) \]

\[ = (\Delta \land (\Gamma \land X)) \lor (\Delta \land (\Gamma \land \neg X)) \]

\[ = \Delta \land (\exists X \Gamma) \]
\text{project}(\Delta + \{O\}, I1..I6)
Minimum Cardinality

\[
\begin{align*}
A \land \text{okX} & \to \neg B \\
\neg A \land \text{okX} & \to B \\
B \land \text{okY} & \to \neg C \\
\neg B \land \text{okY} & \to C
\end{align*}
\]

<table>
<thead>
<tr>
<th>okX</th>
<th>okY</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.</td>
<td>.</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
Minimum Cardinality

A C ~A ~C

~okX ~okY

B ~B

0 0
1 1
2 1

1 1 0 0

~A ~C A C

1 1

~okX ~okY
Minimizing: Requires Smoothness
Minimizing

A. Darwiche
Minimizing
Minimizing

\[ \neg A, B, C, D \]

\[ \neg A \]
\[ B \]
\[ \neg B \]
\[ A \]
\[ C \]
\[ \neg D \]
\[ D \]
\[ \neg C \]
Minimizing

\[ \neg A \land B \land \neg B \land A \lor C \land \neg D \land D \land \neg C \]

A. Darwiche
Minimizing
Minimizing

\[ A \land \neg B \land A \land C \land \neg D \land A, B, \neg C, D \]
## Decomposability

<table>
<thead>
<tr>
<th>Query</th>
<th>DNNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO: Consistency</td>
<td>Yes</td>
</tr>
<tr>
<td>VA: Validity</td>
<td></td>
</tr>
<tr>
<td>CE: Clausal entailment</td>
<td>Yes</td>
</tr>
<tr>
<td>SE: Sentential entailment</td>
<td></td>
</tr>
<tr>
<td>IP: Implicant testing</td>
<td></td>
</tr>
<tr>
<td>EQ: Equivalence testing</td>
<td></td>
</tr>
<tr>
<td>MC: Model Counting</td>
<td></td>
</tr>
<tr>
<td>ME: Model enumeration</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Decomposability

<table>
<thead>
<tr>
<th>Transformation</th>
<th>DNNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD: Conditioning</td>
<td>Yes</td>
</tr>
<tr>
<td>SFO: Single variable</td>
<td>Yes</td>
</tr>
<tr>
<td>FO: Multiple variable</td>
<td>Yes</td>
</tr>
<tr>
<td>&amp;: Conjoin</td>
<td></td>
</tr>
<tr>
<td>B &amp;: Bounded Conjoin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>: Disjoin</td>
</tr>
<tr>
<td>B</td>
<td>: Bounded Disjoin</td>
</tr>
<tr>
<td>~: Negate</td>
<td></td>
</tr>
</tbody>
</table>
Determinism
Determinism
Satisfiability

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C
A, ¬C, okX, okY

Is there a satisfying assignment?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>okX</th>
<th>okY</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Model Counting

Counting Algorithm

How many satisfying assignments?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>okX</th>
<th>okY</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C
A, ¬C, okX, okY
Counting Graph

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{counting_graph.png}
\end{figure}
Counting Models

\[ S = \{ A, \neg B \} \]
Counting Models

\[ S = \{ A, \neg B \} \]
Counting Graph

\[ S = \{ A, \neg B \} \]
Counting Graph

\[ \neg A \quad B \quad \neg B \quad A \quad C \quad \neg D \quad D \quad \neg C \]
$S = \{ A, \neg B \}$
Counting Graph

$S = \{A, \neg B, C\}$

A. Darwiche
Asserting Literals

\[ S = \{ A, \neg B, C \} \cup \{ \neg D \} \]

\[ 1 + (-1)1 = 0 \]
Retracting Literals

\[ S = \{ A, \neg B, C \} \setminus \{ \neg B \} \]

\[ 1 + ( +1 ) 1 = 2 \]
Flipping Literals

\[ S = \{A, \neg B, C\} \setminus \{\neg B\} \cup \{B\} \]

\[ 1 + (+1)1 + (-1)1 = 1 \]
## Testing Equivalence

\[ A \land \text{okX} \rightarrow \neg B \]
\[ \neg A \land \text{okX} \rightarrow B \]
\[ B \land \text{okY} \rightarrow \neg C \]
\[ \neg B \land \text{okY} \rightarrow C \]

<table>
<thead>
<tr>
<th>okX</th>
<th>okY</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Pr(A) = 0.4
Pr(B) = 0.7
Pr(C) = 0.1
Pr(okX) = 0.9
Pr(okY) = 0.8

\[ 0.00864 \]
\[ 0.01944 \]

p
Testing Equivalence

$KB1(A, B, C, okX, okY): p$

$Pr(A) = .50$
$Pr(B) = .05$
$Pr(C) = .74$
$Pr(okX) = .33$
$Pr(okY) = .80$

$KB2(A, B, C, okX, okY): q$

If $p <> q$, then $KB1$ and $KB2$ not equivalent
If $p = q$, then $Pr(KB1$ and $KB2$ are equivalent) $> 1/2$

Run test 100 times $\Rightarrow$ error is $< 10^{-30}$
Probabilistic Equivalence Testing

- Given propositional theories F & G
- Compute p(F), p(G)
- If p(F) ≠ p(G), F & G are not equivalent
- Otherwise, equivalent with probability >1/2
Equivalence Testing

Map x y z to \{1,2,3,4,5,6\}

\[\begin{array}{|c|c|}
\hline
x & y & z \\
\hline
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
\hline
\end{array}\]

\[\begin{align*}
F(xyz) &= (1\cdot2) (1\cdot3) 5 = 10 \\
&= (1\cdot2) 3 (1 - 5) = 12 \\
&= (1\cdot2) 3 5 = -15 \\
&= 2 (1\cdot3) 5 = -20 \\
\end{align*}\]

\[Add: \quad p(F) = -13\]

\[p(F) = p(G) \Rightarrow F \& G \text{ are equivalent with probability } \geq (m – 1)^n / m^n ( > \frac{1}{2} \text{ for } m \geq 2n)\]
Projection under determinism

\begin{align*}
\text{or} & \quad \text{and} \quad \text{or} \\
B & \quad \text{or} & \quad \text{or} & \quad \text{or} & \quad \text{or} & \quad \text{and} \\
\sim A & \quad \sim C & \quad A & \quad C & \quad \sim B \\
\sim \text{ok}X & \quad \sim \text{ok}Y
\end{align*}
Projection under determinism

A C~A
~okX
~okY
true

A Darwiche
## Determinism

<table>
<thead>
<tr>
<th>Query</th>
<th>d-DNNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO: Consistency</td>
<td>Yes</td>
</tr>
<tr>
<td>VA: Validity</td>
<td>Yes</td>
</tr>
<tr>
<td>CE: Clausal entailment</td>
<td>Yes</td>
</tr>
<tr>
<td>SE: Sentential entailment</td>
<td></td>
</tr>
<tr>
<td>IP: Implicant testing</td>
<td>Yes</td>
</tr>
<tr>
<td>EQ: Equivalence testing</td>
<td>?</td>
</tr>
<tr>
<td>MC: Model Counting</td>
<td>Yes</td>
</tr>
<tr>
<td>ME: Model enumeration</td>
<td>Yes</td>
</tr>
</tbody>
</table>

A. Darwiche
## Determinism

<table>
<thead>
<tr>
<th>Transformation</th>
<th>d-DNNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD: Conditioning</td>
<td>Yes</td>
</tr>
<tr>
<td>SFO: Single variable</td>
<td></td>
</tr>
<tr>
<td>FO: Multiple variable</td>
<td></td>
</tr>
<tr>
<td>&amp;: Conjoin</td>
<td></td>
</tr>
<tr>
<td>B&amp;: Bounded Conjoin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>: Disjoin</td>
</tr>
<tr>
<td>B</td>
<td>: Bounded Disjoin</td>
</tr>
<tr>
<td>~: Negate</td>
<td>?</td>
</tr>
</tbody>
</table>
Decision

\[ \begin{align*}
& X_1 \lor \neg X_1 \\
& \quad \lor X_2 \lor \neg X_2 \\
& \quad \quad \lor X_3 \lor \neg X_3 \\
& \quad \quad \quad \lor 1 \lor 0
\end{align*} \]
<table>
<thead>
<tr>
<th>Query</th>
<th>FBDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO: Consistency</td>
<td>Yes</td>
</tr>
<tr>
<td>VA: Validity</td>
<td>Yes</td>
</tr>
<tr>
<td>CE: Clausal entailment</td>
<td>Yes</td>
</tr>
<tr>
<td>SE: Sentential entailment</td>
<td></td>
</tr>
<tr>
<td>IP: Implicant testing</td>
<td>Yes</td>
</tr>
<tr>
<td>EQ: Equivalence testing</td>
<td>?</td>
</tr>
<tr>
<td>MC: Model Counting</td>
<td>Yes</td>
</tr>
<tr>
<td>ME: Model enumeration</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Decision

<table>
<thead>
<tr>
<th>Transformation</th>
<th>FBDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD: Conditioning</td>
<td>Yes</td>
</tr>
<tr>
<td>SFO: Single variable</td>
<td></td>
</tr>
<tr>
<td>FO: Multiple variable</td>
<td></td>
</tr>
<tr>
<td>&amp;: Conjoin</td>
<td></td>
</tr>
<tr>
<td>B&amp;: Bounded Conjoin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>: Disjoin</td>
</tr>
<tr>
<td>B</td>
<td>: Bounded Disjoin</td>
</tr>
<tr>
<td>~: Negate</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Ordering
## Ordering

<table>
<thead>
<tr>
<th>Query</th>
<th>OBDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO: Consistency</td>
<td>Yes</td>
</tr>
<tr>
<td>VA: Validity</td>
<td>Yes</td>
</tr>
<tr>
<td>CE: Clausal entailment</td>
<td>Yes</td>
</tr>
<tr>
<td>SE: Sentential entailment</td>
<td>Yes</td>
</tr>
<tr>
<td>IP: Implicant testing</td>
<td>Yes</td>
</tr>
<tr>
<td>EQ: Equivalence testing</td>
<td>Yes</td>
</tr>
<tr>
<td>MC: Model Counting</td>
<td>Yes</td>
</tr>
<tr>
<td>ME: Model enumeration</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Ordering

<table>
<thead>
<tr>
<th>Transformation</th>
<th>OBDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD: Conditioning</td>
<td>Yes</td>
</tr>
<tr>
<td>SFO: Single variable</td>
<td>Yes</td>
</tr>
<tr>
<td>FO: Multiple variable</td>
<td></td>
</tr>
<tr>
<td>&amp;: Conjoin</td>
<td></td>
</tr>
<tr>
<td>B&amp;: Bounded Conjoin</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>: Disjoin</td>
</tr>
<tr>
<td>B</td>
<td>: Bounded Disjoin</td>
</tr>
<tr>
<td>~: Negate</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Agenda

- Part I: Languages
- Part II: Operations
- Part III: Compilers
- Part IV: Applications
A Knowledge Compilation MAP

Negation Normal Form

Decomposability
Determinism
Smoothness
Flatness
Decision
Ordering

Polytime Operations

Consistency (CO)
Validity (VA)
Clausal entailment (CE)
Sentential entailment (SE)
Implicant testing (IP)
Equivalence testing (EQ)
Model Counting (CT)
Model enumeration (ME)

Projection (exist. quantification)
Conditioning
Conjoin, Disjoin, Negate

Succinctness

A. Darwiche
NNF Subsets

- NNF
  - d-NNF
  - s-NNF
  - DNNF
  - f-NNF

  - BDD
    - FBDD
      - OBDD
  - d-DNNF
    - sd-DNNF
      - MODS
    - EQ?
      - SE,EQ
      - VA,IP,SE,EQ
  - DNF
  - CNF
    - PI
      - VA,IP, SE,EQ
      - CO, CE, ME
      - EQ?
Part III: Compilers
Building Compilers

- To-down approaches:
  - Based on exhaustive search

- Bottom-up approaches:
  - Based on transformations
SAT by DPLL Search

Terminating condition for recursion:
empty set (satisfied), or empty clause (contradiction)

• Unit resolution
• Conflict-directed backtracking
•Clause learning
• Branching heuristics
• Restarts

v = false
Recent Trend: Exhaustive DPLL

- **Count number of models:**
  - Model counters, e.g., relsat, cachet

- **Generate all/subset of models:**
  - Image computation in model checking
  - SMT (non-propositional reasoning)

- **Variations on DPLL Search**
The Language of Search

Exhaustive DPLL

Record Trace

Knowledge Compiler

Variations

Languages

A. Darwiche
Trace of DPLL

\[ X \lor Y \lor \overline{Z} \lor \overline{X} \lor Y \lor \overline{Z} \]

\[ X \lor \overline{Y} \lor \overline{Z} \]

A. Darwiche
Exhaustive DPLL

Run to Exhaustion

\[ X \lor Y \]
\[ X \lor \neg Y \lor \neg Z \]
\[ \neg X \lor Y \lor \neg Z \]
Trace of DPLL: a Formula

A. Darwiche
Trace of DPLL: a Formula

Equivalent to original CNF

Tractable
(e.g., count models)
Dealing with Redundancy

**Level One:** Do not record redundant portions of trace

**Level Two:** Try not to solve equivalent subproblems
Dealing with Redundancy
Dealing with Redundancy

Simply create
existing node

A. Darwiche
This is an OBDD!
This is an OBDD!

NNF + decision, decomposability, ordering
A Non-traditional OBDD Compiler

Compile

Exhaustive DPLL, Fixed variable order, Unique nodes

New complexity guarantees
FBDD

Exhaustive DPLL, Dynamic variable order, Unique nodes

Compile

NNF + decision, decomposability

A. Darwiche
FBDD vs OBDD

- FBDD more succinct than OBDD
  \textit{(dynamic var ordering in SAT)}

- OBDD: equivalence test (canonical)
- FBDD: probabilistic equivalence test
- Both allow model counting
Dealing with Redundancy

- Level One: Unique nodes (done)
- Level Two: Avoid redundant compilation (search)
Redundant Compilation

\[
\begin{align*}
&x_5 \lor x_6 \\
&x_4 \lor \neg x_5 \lor x_6 \\
&x_1 \lor x_3 \lor x_4 \lor x_5 \\
&x_2 \lor x_3 \\
&x_1 \lor x_2 \lor \neg x_3
\end{align*}
\]

Formula Caching: complexity guarantees
Formula Caching

- Majercik and Litmman, 1998
- Darwiche, 2002
- Bacchus et al, 2003, 2004
- Huang & Darwiche, 2004
- Sang, Kautz, Beam, 2004, 2005
- Thurley, 2006
Caching for DPLL

\[ \text{OBDD}(\Delta) = v_1 \]

\[ \text{OBDD}(\Delta |_{v_1=0}) \quad \text{OBDD}(\Delta |_{v_1=1}) \]

\[ \text{OBDD}(\Delta |_{v_1=0, v_2=0}) \quad \text{OBDD}(\Delta |_{v_1=1, v_2=1}) \]

Recursive calls may be made on equivalent CNFs
Caching for DPLL

\[ \Delta = \{ v_5 + v_6 \} \]

\[ v_4 + \neg v_5 + v_6 \]
\[ v_1 + v_3 + v_4 + v_5 \]
\[ v_2 + v_3 \]
\[ v_1 + v_2 + \neg v_3 \} \]

OBDD(\( \Delta \))

\[
\begin{array}{c|c}
\text{\( v_1 v_2 v_3 \)} & \Delta' \\
\hline
0 0 0 & \text{contradiction} \\
0 0 1 & \text{contradiction} \\
0 1 0 & v_5 + v_6, v_4 + v_5 \\
0 1 1 & v_5 + v_6, v_4 + v_5 \\
1 0 0 & \text{contradiction} \\
1 0 1 & v_5 + v_6, v_4 + v_5 \\
1 1 0 & v_5 + v_6, v_4 + v_5 \\
1 1 1 & v_5 + v_6, v_4 + v_5 \\
\end{array}
\]

A. Darwiche
Caching for DPLL

- After instantiation of \( v_1v_2v_3 \), \( \Delta' \) is either contradictory, or determined by clause \( c_3 \) alone.
- \( c_3 \) can only be in one of two states: satisfied or shrunk to \( v_4v_5 \)
Caching for Basic DPLL

- In general, cutset\(_i\) is set of clauses mentioning a variable \(\leq v_i\) and one \(> v_i\).
- After instantiation of \(v_1v_2...v_i\), \(\Delta'\) is either contradictory, or determined by states of clauses cutset\(_i\).
- Number of distinct \(\Delta'\) is \(\leq 2 |\text{cutset}_i| + 1\).
- Maintain a cache for each \(i\), and use the value of cutset\(_i\)—a bit vector—as key.
CNF to OBDD

OBDD(Δ, i){
    if(contradiction) return 0-sink
    if(satisfied) return 1-sink
    key = value(cutset^{i-1})
    lookup = cache_{i-1}[key]
    if(lookup ≠ nil) return lookup
    result = getnode_node(v_i, OBDD(Δ|v_{i=0}, i+1), OBDD(Δ|v_{i=1}, i+1))
    cache_{i-1}[key] = result
    return result
}
Complexity

- For each $i$, $2^{|cutset_i|}$ bounds
  - number of recursive calls $OBDD(\Delta, i+1)$
  - number of entries in $cache_i$
  - number of OBDD nodes labeled with $v_i$
- Size of largest cutset is known as *cutwidth* of variable order
- Time and space complexities of algorithm and size of OBDD are all linear in number of variables, and exponential only in cutwidth
- Variable orders with small cutwidth can help
Complexity Theorems

- \(\text{size}(\text{OBDD}) \leq n2^w + 2\)
  - \(n\): number of variables
  - \(w\): cutwidth of variable order (size of largest cutset)

- Time complexity = \(O(sn2^w)\)
  - \(s\): size of CNF

- Also hold for \(w = \text{pathwidth}\), using a slightly different caching scheme

- Cutwidth and pathwidth are incomparable
Beyond BDDs...

Plain DPLL $\rightarrow$ FBDD

Fixed Variable Ordering $\rightarrow$ OBDD
Decomposition (Component Analysis)

Solve disjoint subproblems independently

Combine as AND node
d-DNNF
Deterministic Decomposable Negation Norm Form (d-DNNF)

\[
\begin{align*}
    &A \lor B \lor C \quad \neg A \lor \neg B \lor C \\
    &A \lor D \lor E \quad \neg A \lor \neg D \lor E
\end{align*}
\]
Deterministic Decomposable Negation Norm Form (d-DNNF)
Decomposition Methods

- Dynamically detect disjoint components
  - Most effective, but very expensive

- Static structural analysis
  - Constructs a decomposition tree (dtree)
  - Does not detect all decompositions
  - Low overhead at runtime
FBDD vs d-DNNF

- d-DNNF more succinct than FBDD
  \textit{(effectiveness of decomposition)}

- Deterministic equivalence test open
- Probabilistic equivalence test applies
- Other queries same...
The Language of Search

- Fixed Variable Ordering → OBDD
- Plain DPLL → FBDD
- Allowing Decomposition → d-DNNF

Other languages: deterministic DNF

A. Darwiche
Relation to AND/OR Search (CP)

- AND/OR graphs are deterministic and decomposable.

- AND/OR search algorithms are doing enough work to compile networks into (multi-valued equivalent of) d-DNNF.

- Capable of more than answering a single query (model counting, belief revision, etc).
Implications

- SAT techniques harnessed for knowledge compilation
  - c2d compiler based on Rsat Solver (SAT Competition 2007): uses dtrees

- Language properties (succinctness/tractability) help characterize power and limitations of search
Understanding DPLL

Take any program \( \mathbf{X} \) that runs exhaustive DPLL-style search.

Examine traces, if traces \( \subseteq \mathcal{L} \), then

- \( \mathbf{X} \) can answer all queries tractable for \( \mathcal{L} \)
- \( \mathbf{X} \) is hopeless on any input having no polynomial-size representation in \( \mathcal{L} \)

A. Darwiche
Traces of several model counters (Relsat, Cachet, e.g.) are in $\text{d-DNNF}$

Are doing enough work to

- compile formulas into $\text{d-DNNF}$
- solve tasks beyond model counting (e.g., minimum cardinality, probabilistic equivalent testing)
Limitation of DPLL: General Determinism

Decision nodes (d-DNNF’)

Deterministic nodes (d-DNNF)
Beyond DPLL: Decomposability (D) Without Determinism (d)

DNNF: CO, CE, ME, exist quant

A. Darwiche
\[ \Delta = \Delta^l \cup \Delta^r \]

\[ \text{and} \]

\[ \text{dnnf}(\Delta^l) \quad \text{dnnf}(\Delta^r) \]

\[ XY \quad XY \]
Approximate Compilation

\[ \Delta = \Delta^l \cup \Delta^r \]

Diagram:
- \( \Delta \)
  - \( \Delta^l \)
    - \( \text{dnnf}(\Delta^l)_{XY} \)
    - \( \text{dnnf}(\Delta^r)_{XY} \)
  - \( \Delta^r \)
    - \( \text{or} \)
      - \( \text{and} \)
        - \( XY \)
          - \( \text{dnnf}(\Delta^l)_{XY} \)
          - \( \text{dnnf}(\Delta^r)_{XY} \)
      - \( \text{and} \)
        - \( \sim XY \)
        - \( X \sim Y \)
      - \( \text{and} \)
        - \( \sim X \sim Y \)

A. Darwiche
Approximate Compilation

\[ \Gamma \models \Delta = \Delta^l \cup \Delta^r \]

Sound, but not complete
\[ \Gamma \models \Delta = \Delta^l \cup \Delta^r \equiv \Sigma \]

**Approximate Compilation**

\[
\begin{align*}
&\text{or} \\
&\begin{align*}
&\text{and} \\
&X \quad \text{dnnf}(\Delta^l|X) \\
&\text{dnnf}(\Delta^r|X)
\end{align*} \\
&\begin{align*}
&\text{and} \\
&\neg X \quad \text{dnnf}(\Delta^l|\neg X) \\
&\text{dnnf}(\Delta^r|\neg X)
\end{align*}
\end{align*}
\]

Complete, but not sound

A. Darwiche
Bottom-up Compilation
Bottom-up OBDD Construction

CNF: \((x + y) (y + z)\)
Variable order: \(x, y, z\)

The \textit{Apply} algorithm:
combinations two OBDDs using any one of the 16 binary Boolean operators
Bottom-Up OBDD Construction

- OBDD packages, such as CUDD
  - implement $Apply$ (conjoin, disjoin, etc)
  - garbage-collect dead nodes
- $Apply$ is efficient: quadratic in operand size
- Problem: intermediate OBDDs can be much larger than final one—many dead nodes
- uf100-08 (32 models): OBDD has 176 nodes under MINCE order; 30,640,582 intermediate nodes using CUDD; taking 25 mins
DPLL Based Construction

\[ \Delta = (x + y) (y + z) \]

\[ \Delta|_{x=0} = y \]

\[ \Delta|_{x=1} = y + z \]
Missing Opportunities

- Bottom up construction methods for DNNF and d-DNNF
Agenda

- Part I: Languages
- Part II: Operations
- Part III: Compilers
- Part IV: Applications
Part IV: Applications
Applications

- Model-based diagnosis
- Planning
- Probabilistic reasoning
Model-based Diagnosis

System model $\Delta$:  
- okX $\rightarrow$ (A $\leftrightarrow$ $\neg$C)  
- okY $\rightarrow$ (B $\land$ C) $\leftrightarrow$ D  

Health variables: okX, okY
Observables: A, B, D
Nonobservable: C
Model-based Diagnosis

System model $\Delta$:
- $\text{okX} \rightarrow (A \leftrightarrow \neg C)$
- $\text{okY} \rightarrow (B \land C) \leftrightarrow D$

Diagnosis: Values of $(\text{okX}, \text{okY})$ consistent with $\Delta \land \alpha$:
- $(0, 0), (0, 1), (1, 0)$

Abnormal observation $\alpha$:
- $\neg A \land B \land \neg D$

A. Darwiche
Model-based Diagnosis

Abnormal observation $\alpha$: $\neg A \land B \land \neg D$

System model $\Delta$:
- $\text{okX} \rightarrow (A \leftrightarrow \neg C)$
- $\text{okY} \rightarrow (B \land C) \leftrightarrow D$

Minimum cardinality Diagnoses:
- $(0, 0)$, $(0, 1)$, $(1, 0)$
Model-based Diagnosis

Methods for characterizing diagnoses:
- Conflicts
- Kernel diagnoses

Methods for manipulating diagnoses:
- Find minimal diagnoses
- Find minimum-cardinality diagnoses
- Characterize lost functionality
Characterizing Diagnoses

Diagram:

- A
- C
- B
- D
- E
- 1
- 2
- 3
- 4
- 5
- F
- G

A. Darwiche
Characterizing Diagnoses

A

C

E

B

D

1 2 3

4 5

F G

A. Darwiche
Characterizing Diagnoses
Minimal Conflicts

\[ (\neg ok_1 \lor \neg ok_2 \lor \neg ok_4 ), \]
\[ (\neg ok_1 \lor \neg ok_3 \lor \neg ok_4 \lor \neg ok_5 ) \]

An instantiation of ok1,...ok5 is a diagnosis iff it is consistent with (conjunction of) minimal conflicts
Kernel Diagnoses

\[ \neg ok_1, (\neg ok_2 \land \neg ok_3), \]
\[ (\neg ok_2 \land \neg ok_5), \neg ok_4 \]

An instantiation of ok1,...ok5 is a diagnosis iff it is consistent with (disjunction of) kernel diagnoses
Characterizing Diagnosis

- Device Model \( \Delta \)
- Observables \( O_1, \ldots, O_n \)
- Health Variables \( H_1, \ldots, H_m \)
- Others \( X_1, \ldots, X_k \)

\[
\exists X_1, \ldots, X_k, O_1, \ldots, O_n (\Delta \land \alpha)
\]
OBDD
DNNF

\( \neg ok_1 \land \neg ok_4 \land \neg ok_2 \lor ok_1 \lor ok_4 \land \neg ok_3 \land \neg ok_5 \)
Minimal Conflicts

\[ (\neg ok_1 \lor \neg ok_2 \lor \neg ok_4 ) \land \\
(\neg ok_1 \lor \neg ok_3 \lor \neg ok_4 \lor \neg ok_5 ) \]
Kernel Diagnoses

\[
\neg ok_1 \lor (\neg ok_2 \land \neg ok_3) \lor \\
(\neg ok_2 \land \neg ok_5) \lor \neg ok_4
\]
Characterizing Diagnosis

- Device Model
- Observables $O_1, \ldots, O_n$
- Health Variables $H_1, \ldots, H_m$
- Others $X_1, \ldots, X_k$

Health Condition:

$$\exists X_1, \ldots, X_k, O_1, \ldots, O_n (\Delta \land \alpha)$$
System Model

Compile

Succinctness

Efficient computation of diagnoses
System Model

Compile

OBDD vs. DNNF

Succinctness

Efficient computation of diagnoses
Compiling System Models

- DNNF is more succinct: can lead to smaller compilation
- Smaller compilation of system model does not imply faster on-line diagnosis
- Although less succinct, OBDD is more powerful (DNNF is a weaker form)
- Is this extra power relevant?
Diagnosis Using DNNF

Observation: \( \neg A \land B \land \neg D \)

System model \( \Delta \):
\begin{align*}
\text{okX} &\rightarrow (A \leftrightarrow \neg C) \\
\text{okY} &\rightarrow (B \land C) \leftrightarrow D
\end{align*}
Diagnosis Using DNNF

Observation: \( \neg A \land B \land \neg D \)

System model \( \Delta \):

- \( \text{okX} \rightarrow (A \leftrightarrow \neg C) \)
- \( \text{okY} \rightarrow (B \land C) \leftrightarrow D \)
Diagnosis Using DNNF

Set observables
- Linear time

Project out nonobservables
- Linear time

Projection exponential for OBDD
(can be made linear with particular orders)
Minimum Cardinality Diagnoses

- Number of diagnoses can be large
- Some may be more preferable than others
- Minimize number of faulty components in diagnosis
- Again, easy for DNNF
Minimum Cardinality Diagnoses

Smoothing: make disjuncts mention same set of variables; $O(|\Delta'| \cdot |H|)$
Minimizing the DNNF
Minimum Cardinality Diagnoses

or

and

¬okX

okY

okX

¬okY
Minimum Cardinality Diagnoses Using OBDDs

- Create “filter” OBDD that asserts a given cardinality $k$
- Conjoin it with OBDD that represents set of diagnoses
- Repeat for $k = 0, 1, 2...$ until result is nonempty
## Comparison

<table>
<thead>
<tr>
<th><strong>Operation</strong></th>
<th><strong>OBDD</strong></th>
<th><strong>DNNF</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition($\Delta'$, $\alpha$)</td>
<td>$O(</td>
<td>\Delta'</td>
</tr>
<tr>
<td>Project($\Gamma$, $H$)</td>
<td>exponential</td>
<td>$O(</td>
</tr>
<tr>
<td>Minimize($\Delta_d$)</td>
<td>$O(mc \cdot</td>
<td>\Delta_d</td>
</tr>
</tbody>
</table>
Characterizing Lost Functionality

\[
\text{Minimize}(\text{project}(\Delta \land \alpha, H_1 \ldots H_m)) \land \Delta
\]
Hierarchical Diagnosis
Scalability

- Requires a health variable for each component

- c1908 has 880 gates; basic encoding fails to compile

- New technique to reduce number of health variables

- Preserves soundness and completeness w.r.t. min-cardinality diagnoses

- Requires only 160 health variables for c1908
Hierarchical Diagnosis
Hierarchical Diagnosis

A. Darwiche
Hierarchical Diagnosis

A. Darwiche
Identifying Cones

- Gate $G$ dominates gate $X$ if any path from $X$ to output of circuit contains $G$

- All gates dominated by $G$ form a cone

- Dominators found by breath-first traversal of circuit

- Treat maximal cones as blackboxes
Abstraction of Circuit

$\Theta_C = \{T, U, V, A, B, C\}$
Top-level Diagnosis

Diagnosis: \{A, B, C\}
Diagnosis of Cone

- Need to set inputs/output of cone according to top-level diagnosis

- Rest is similar, but not a simple recursive call (to avoid redundancy)

- Once cone diagnoses found, global diagnoses obtained by substitution
Diagnosis of Cone

Top-level diagnosis:  
\{A, B, C\}

3 diagnoses for cone A:  
\{A\}, \{D\}, \{E\}

3 global diagnoses by substitution:  
\{A, B, C\}  
\{D, B, C\}  
\{E, B, C\}
Soundness

- Top-level diagnoses have same cardinality. Substitutions do not alter cardinality (cones do not overlap).

- Remains to show that cardinality of these diagnoses, \( d \), is smallest. Proof by contradiction:

- Suppose there is diagnosis \( |P| < d \). Replace every gate in \( P \) with its highest dominator to obtain \( P' \).

- \( P' \) is a valid top-level diagnosis, contradicting soundness of baseline diagnoser.
Completeness

- Need to show every min-cardinality diagnosis is found.

- Given diagnosis $P$ of min cardinality $d$, replace every gate in $P$ with its highest dominator to obtain $P'$.

- $P'$ has cardinality $d$, and only mentions gates in top-level abstraction, and hence will be found by top-level diagnosis (by completeness of baseline diagnoser).

- $P$ itself will be found by substitution (by completeness of cone diagnosis).
Sequential Diagnosis
Sequential Diagnosis

- Set of min-cardinality diagnoses may still be large
- Faults not identified with certainty
- Take measurements, one at a time, until faults identified
- Would like to minimize number of measurements
- Use heuristic based on amount of information gain
Sequential Diagnosis

- Assume a probabilistic model of the system
  - Failure probabilities for components
  - Output behavior of faulty component (e.g., 1 and 0 with equal probability)
  - These implicitly define a joint probability distribution over all system variables

- Encoding and compilation described later

- Pick component with highest posterior probability of failure, measure variable with highest entropy in that component
Hierarchical Sequential Diagnosis

1. To improve scalability, previous idea of abstraction applies.
2. Treat each cone (blackbox) as a single “big” component.
3. Need to compute a single failure probability that “summarizes” the failure behavior of the cone.
4. Create copy of cone with all gates healthy, feed outputs of two cones into XOR gate, compute \( \Pr(\text{output} = 1) \).
Planning
Slippery Gripper

Goal: block painted and held, gripper clean

Probabilistic action effects:
- Paint: paints block w. p. 1; makes gripper dirty w. p. 1 if it holds block, w. p. 0.1 if not
- Pick-up: succeeds w. p. 0.5 if gripper wet, 0.95 if gripper dry
- Dry: dries wet gripper w. p. 0.8; doesn’t affect dry gripper

Probabilistic initial state:
- block not painted, not held
- gripper clean, but dry with probability 0.7
Conformant Probabilistic Planning

- Probabilistic initial state and action effects

- Conformant: action effects not observable
  - Can’t decide next action by observing environment

- Needs straight-line plan with max probability of success, for given horizon (number of steps)

- Example 2-step plan: [paint, pickup] (succeeds with probability 0.7335)
Brute-force Approach

- Compute success probability for all plans of length one

- Given success probabilities of plans of length $i$, compute probability of success for plans of length $i + 1$

- Iterate to planning horizon $n$

- Pick $n$-step plan with max success probability

- Exponential in planning horizon

A. Darwiche
Why is it hard?

- Consider decision version
  - Does there exist plan of success probability > p, for given horizon

- Given plan, deciding if success probability > p is PP-complete
  - Can be reduced to MAJ-SAT: does majority of assignments satisfy CNF
  - Alternatively: Is CNF satisfiable with probability > ½

- Finding plan with given property (that is free to test) is NP-complete, for given horizon

- Conformant probabilistic planning for given horizon is \( \text{NP}^{\text{pp}} \)-complete

- \( \text{NP} \subseteq \text{PP} \subseteq \text{NP}^{\text{pp}} \)
Propositional Encoding: Initial State

- State space: **BP** (block-painted), **BH** (block-held), **GC** (gripper-clean), **GD** (gripper-dry)

- Probabilistic initial state: \{\neg BP, \neg BH, GC, \ p \leftrightarrow GD\}
  - Chance variable: \ p \ (labeled with 0.7)
  - \ p = 1: \ {\neg BP, \neg BH, GC, GD} \ probability 0.7
  - \ p = 0: \ {\neg BP, \neg BH, GC, \neg GD} \ probability (1 - 0.7)

- Encoded as propositional sentence; some variables labeled with numbers (probabilities)
### Propositional Encoding: Action Effects

- **Dry**: dries wet gripper w. p. 0.8; doesn’t affect dry gripper
  - \( \neg \text{dry} \vee \text{GD} \vee (q \leftrightarrow \text{GD}'), \neg \text{dry} \vee \neg \text{GD} \vee \text{GD}' \)

- **Frame axiom**
  - \( \neg \text{dry} \vee (\text{BP} \leftrightarrow \text{BP'}), \neg \text{dry} \vee (\text{BH} \leftrightarrow \text{BH'}), \neg \text{dry} \vee (\text{GC} \leftrightarrow \text{GC'}) \)

- Each setting of chance variable selects one effect
  - probability of effect is label of chance variable, or 1 minus that depending on value of variable
  - multiple chance variables: multiply the numbers

- Define \( \text{Pr}(e) \) where \( e \) is an *event* (setting of chance vars)
Propositional Encoding: Goal

- Goal: \{BP', BH', GC'\}
Propositional Encoding: Horizon n

- State variables: $S_0, S_1, S_2, \ldots, S_n$
- Chance variables: $P_{-1}, P_0, P_1, \ldots, P_{n-1}$
- Action variables: $A_0, A_1, A_2, \ldots, A_{n-1}$

- Initial state: $I(P_{-1}, S_0)$  
  Goal: $G(S_n)$

- Plan step: $A_k \equiv A(S_k, A_k, P_k, S_{k+1})$

- $\Delta_n \equiv I(P_{-1}, S_0) \land A_0 \land A_1 \land \ldots \land A_{n-1} \land G(S_n)$
Plan Assessment

- Given propositional encoding $\Delta_n$ and n-step plan $\pi$

- What’s probability of success $\Pr(\Delta_n, \pi)$?

- Plan $\pi$ is instantiation of action vars

- $\Pr(\Delta_n, \pi)$ is sum of $\Pr(e)$ for $e$ consistent with $\Delta_n \land \pi$

- Computation intractable (has to enumerate models)
Brute-force Algorithm

- Search through all plans $\pi$ (instantiations of action vars)
  - For each plan $\pi$, compute $Pr(\Delta_n, \pi)$
  - Return plan $\pi$ with max $Pr(\Delta_n, \pi)$
- Improve in two ways
  - Efficient plan assessment
  - Search space pruning
- Both achieved by compiling $\Delta_n$ to d-DNNF
Compilation to d-DNNF

- Use existing compiler c2d, http://reasoning.cs.ucla.edu/c2d/
- Compilation intractable in general
- For structured problems, exponential only in treewidth
- Treewidth does not grow with horizon in this encoding
- Can scale to large horizons
d-DNNF

\[
\begin{align*}
&\text{and} \\
&\text{or} \\
&\text{and} \\
&t' \quad \neg p \\
&\neg p \\
&s' \\
&\neg \text{dry} \\
&\neg \text{paint} \\
&\neg \text{pickup} \\
&\neg r \\
&\neg \text{dry}' \\
&\neg \text{paint}' \\
&\text{pickup}'
\end{align*}
\]
Plan Assessment on d-DNNF

[paint, pickup']

A. Darwiche
Plan Assessment on d-DNNF

[paint, pickup']

A. Darwiche
Plan Assessment on d-DNNF

[Paint, pickup']

A. Darwiche
Plan Assessment on d-DNNF

[paint, pickup']

A. Darwiche
Plan Assessment on d-DNNF

[paint, pickup']

or

and

and

and

t' ¬p p s' ¬dry paint ¬pickup ¬r ¬dry' ¬paint' pickup'
Plan Assessment on d-DNNF


dry paint pickup

A. Darwiche
Plan Assessment on d-DNNF

- Set action vars to constants according to plan $\pi$
- Set all state vars to 1 (existential quantification)
- Turn chance vars and their negations into numbers
- Turn or into summation, and into multiplication
- Evaluate resulting arithmetic circuit
- Value at root is $\Pr(\Delta_n, \pi)$
Assessment of Partial Plan

- Partial plan $\pi$: instantiation of a subset of action vars
- $\Pr(\Delta_n, \pi)$: success probability of best completion of $\pi$
- Maximize over uninstantiated action vars
- Special d-DNNF:
  
  $\begin{array}{c}
  \text{or} \\
  \text{and} & \text{and} \\
  \beta & \neg x & x & \gamma
  \end{array}$
- Turn corresponding or node into max

A. Darwiche
Assessment of Partial Plan

- Summations over state vars, followed by maximizations over action vars

- All max must be performed after sum

- Not guaranteed in d-DNNF

- Max and sum nodes mixed in any order

- Some max performed too early: Result incorrect!

A. Darwiche
Depth-first Branch-and-Bound

- Key observation: performing max too early can only increase result

- Result is upper bound on true value

- Partial plan can be pruned if upper bound $\leq$ value of best plan already found

- Depth-first branch-and-bound: will find optimal plan

- Tighter bounds leads to more pruning
Plan Search

A. Darwiche
Plan Search

T 0.151

C

B

tcb 0.009
tcb' 0.079
tc'b 0.117
tc'b' 0.053

C

B

t'cb 0.053
t'cb' 0.023
t'c'b 0.051
t'c'b' 0.031

Best Score: 0

A. Darwiche
Plan Search

Best Score: 0

T  .151

C  .135

B  .009  .079  .117  .053  .023  .051  .031  .055

A. Darwiche
Plan Search

T \cdot 0.151

C \cdot 0.135

B \cdot 0.127

B \cdot \begin{cases} tc'b \cdot 0.023 \\ t'c'b \cdot 0.051 \\ t'c'b' \cdot 0.031 \end{cases}

C \cdot \begin{cases} tcb \cdot 0.009 \\ tcb' \cdot 0.079 \\ tc'b \cdot 0.117 \\ tc'b' \cdot 0.053 \end{cases}

Best Score: 0
Plan Search

Best Score: 0.009

T 0.151

C 0.135

C

B 0.127

B

B

B

B

B

tcb

tcb'

tc'b

tc'b'

t'cb

t'cb'

t'c'b

t'c'b'

0.009

0.079

0.117

0.053

0.023

0.051

0.031

0.055

A. Darwiche
Plan Search

Best Score: 0.079
Plan Search

Best Score: 0.079

T 0.151

C 0.135

B

B 0.120

B

B

Tc'b 0.009
tc'b' 0.079
tc'b 0.117
tc'b' 0.053
t'cb 0.023
t'cb' 0.051
tc'b 0.031
tc'b 0.055
Plan Search

Best Score: 0.117

T 0.151

C 0.135

B 0.120

B 0.079 0.117 0.053 0.023 0.051 0.031 0.055
Plan Search

Best Score: 0.117

T.151

C

Tcb

Tc′b

Tcb′

Tc′b′

B

B

B

B

Tcb

Tc′b

Tcb′

Tc′b′

0.009

0.079

0.117

0.053

0.023

0.051

0.031

0.055
Plan Search

T \[ T \]

Best Score: \[ .117 \]

A. Darwiche
Probabilistic Reasoning

A. Darwiche
Bayesian Networks

If Battery Power = OK, then Lights = ON (99%) …

<table>
<thead>
<tr>
<th>Battery Power</th>
<th>Lights</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>ON</td>
<td>.99</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>.01</td>
</tr>
<tr>
<td>WEAK</td>
<td>ON</td>
<td>.20</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>.80</td>
</tr>
<tr>
<td>DEAD</td>
<td>ON</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>1</td>
</tr>
</tbody>
</table>

A. Darwiche
\[
\begin{array}{cccc}
A & B & C & \text{Pr(.)} \\
\theta_a & \theta_b|a & \theta_c|a & \theta_a \theta_b|a \theta_c|a \\
\theta_a & \theta_b|a & \theta_{\sim c}|a & \theta_a \theta_b|a \theta_{\sim c}|a \\
\theta_a & \theta_{\sim b}|a & \theta_c|a & \theta_a \theta_{\sim b}|a \theta_c|a \\
\theta_a & \theta_{\sim b}|a & \theta_{\sim c}|a & \theta_a \theta_{\sim b}|a \theta_{\sim c}|a \\
\cdot & \cdot & \cdot & \cdot \cdot \cdot \cdot \\
\end{array}
\]
Joint Distributions

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>Pr(.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true</td>
<td>true</td>
<td>.03</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td>.27</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>true</td>
<td>.56</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>false</td>
<td>.14</td>
</tr>
</tbody>
</table>

\[
\Pr(a) = .03 + .27 = .3
\]
Joint Distributions

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Pr(.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>.03</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>.27</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>.56</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>.14</td>
</tr>
</tbody>
</table>

\[
\Pr(\neg b) = .27 + .14 = .41
\]
Joint Distributions

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Pr(.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>$\lambda_a \lambda_b * .03$</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>$\lambda_a \lambda_{\neg b} * .27$</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>$\lambda_{\neg a} \lambda_b * .56$</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>$\lambda_{\neg a} \lambda_{\neg b} * .14$</td>
</tr>
</tbody>
</table>

$F(\lambda_{\neg a}, \lambda_{\neg b}, \lambda_a, \lambda_b) =
\.03\lambda_a \lambda_b + .27\lambda_a \lambda_{\neg b} + .56\lambda_{\neg a} \lambda_b + .14\lambda_{\neg a} \lambda_{\neg b}$
\[ F(\lambda_{\sim a}, \lambda_{\sim b}, \lambda_a, \lambda_b) \]
\[ = 0.03\lambda_a \lambda_b + 0.27\lambda_a \lambda_{\sim b} + 0.56\lambda_{\sim a} \lambda_b + 0.14\lambda_{\sim a} \lambda_{\sim b} \]

\[ \Pr(a, \sim b) \]
\[ = F(\lambda_{\sim a}:0, \lambda_{\sim b}:1, \lambda_a:1, \lambda_b:0) \]
\[ = 0.27 \]

\[ \Pr(a) \]
\[ = F(\lambda_{\sim a}:0, \lambda_{\sim b}:1, \lambda_a:1, \lambda_b:1) \]
\[ = 0.03 + 0.27 \]
\[
\begin{align*}
A & \quad B & \quad C & \quad \text{Pr}(.) \\
\begin{array}{cccc}
a & b & c & \theta_a \theta_b|a \theta_c|a \\
a & b & \sim c & \theta_a \theta_b|a \theta_{\sim c}|a \\
a & \sim b & c & \theta_a \theta_{\sim b}|a \theta_c|a \\
a & \sim b & \sim c & \theta_a \theta_{\sim b}|a \theta_{\sim c}|a \\
. & . & . & . \\
. & . & . & \ldots
\end{array}
\end{align*}
\]
\[ A \xrightarrow{\theta_a} B \xrightarrow{\theta_{b|a}} C \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Pr(.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>( \lambda_a \lambda_b \lambda_c \theta_a \theta_{b</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>(\sim c)</td>
<td>( \lambda_a \lambda_b \lambda_{\sim c} \theta_a \theta_{b</td>
</tr>
<tr>
<td>a</td>
<td>(\sim b)</td>
<td>c</td>
<td>( \lambda_a \lambda_{\sim b} \lambda_c \theta_a \theta_{\sim b</td>
</tr>
<tr>
<td>a</td>
<td>(\sim b)</td>
<td>(\sim c)</td>
<td>( \lambda_a \lambda_{\sim b} \lambda_{\sim c} \theta_a \theta_{\sim b</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>...</td>
</tr>
</tbody>
</table>
\[ F = \lambda_a \lambda_b \lambda_c \theta_a \theta_{bla} \theta_{cla} + \]
\[ \lambda_a \lambda_b \lambda_{\sim c} \theta_a \theta_{bla} \theta_{\sim cla} + \]
\[ \lambda_a \lambda_{\sim b} \lambda_c \theta_a \theta_{\sim bla} \theta_{cla} + \]
\[ \lambda_a \lambda_{\sim b} \lambda_{\sim c} \theta_a \theta_{\sim bla} \theta_{\sim cla} \]
\[ \ldots \]
Each term has $2n$ variables ($n$ indicators, $n$ parameters)
Each variable has degree one (multi-linear function)
Multi-Linear Functions ➔ Arithmetic Circuits

\[ f = \lambda_a \lambda_b \theta_a \theta_{b\bar{a}} + \lambda_a \lambda_b \theta_{\bar{a}} \theta_{b\bar{a}} + \lambda_a \lambda_b \theta_{\bar{a}} \theta_{b\bar{a}} + \lambda_{\bar{a}} \lambda_b \theta_{\bar{a}} \theta_{b\bar{a}} \]

A. Darwiche
Reduction to Logic

CNF → Compiler: http://reasoning.cs.ucla.edu/c2d → d-DNNF

Encode

Multi-Linear Function

Arithmetic Circuit

Decode
MLFs $\rightarrow$ ACs

CNFs $\rightarrow$ d-DNNF

Propositional theory: $c \land (a \lor \neg b)$

Multi-linear function: $a \ c + a \ b \ c + c$

Smooth d-DNNF

Arithmetic Circuit

A. Darwiche
Propositional Encoding of Multi-Linear Functions

Propositional theory:
\[ \Delta = c \land (a \lor \neg b) \]

Encodes:
\[ F = a \cdot c + a \cdot b \cdot c + c \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Encodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>abc</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>ab</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>ac</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>a</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>bc</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>b</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>c</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

A. Darwiche
Encoding Network as CNF

\[ F = \lambda_a \lambda_b \theta_a \theta_{bla} + \lambda_a \lambda_{\neg b} \theta_a \theta_{b|a} + \lambda_{\neg a} \lambda_b \theta_{\neg a} \theta_{b|a} + \lambda_{\neg a} \lambda_{\neg b} \theta_{\neg a} \theta_{b|a} \]

\[ \lambda_a \lor \lambda_{\neg a} \quad \neg \lambda_a \lor \neg \lambda_{\neg a} \]
\[ \lambda_b \lor \lambda_{\neg b} \quad \neg \lambda_b \lor \neg \lambda_{\neg b} \]
\[ \lambda_a \leftrightarrow \theta_a \quad \lambda_{\neg a} \leftrightarrow \theta_{\neg a} \]
\[ \lambda_a \land \lambda_b \leftrightarrow \theta_{b|a} \quad \lambda_a \land \lambda_{\neg b} \leftrightarrow \theta_{\neg b|a} \quad \lambda_{\neg a} \land \lambda_b \leftrightarrow \theta_{b|a} \quad \lambda_{\neg a} \land \lambda_{\neg b} \leftrightarrow \theta_{\neg b|a} \]

A. Darwiche
Why Logic?

- Encoding local structure is easy:
  - Determinism encoded by adding clauses:
    \[ \theta_{C \mid A} = 0 \]
  - CSI encoded by collapsing variables:
    \[ \theta_{C \mid AB} = \theta_{C \mid A \neg B} \]
Global Structure: Treewidth \( w \)

\[ O(n \exp(w)) \]
Local Structure: CSI and Determinism
Local Structure: CSI and Determinism
Local Structure: CSI and Determinism

If Battery Power = Dead, then Lights = OFF

<table>
<thead>
<tr>
<th>Lights</th>
<th>Battery Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>0.9</td>
</tr>
<tr>
<td>OFF</td>
<td>0.01</td>
</tr>
<tr>
<td>DEAD</td>
<td>0.80</td>
</tr>
<tr>
<td>DEAD</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Determinism

A. Darwiche
Local Structure

- Functional constraints
- Context-specific independence

A. Darwiche

**Tabular CPT**

| A  | B  | C  | Pr(S|A,B,C) |
|----|----|----|------------|
| a  | b  | c  | 0.95       |
| a  | b  | c  | 0.95       |
| a  | b  | c  | 0.20       |
| a  | b  | c  | 0.05       |
| a  | b  | c  | 0.00       |
| a  | b  | c  | 0.00       |
| a  | b  | c  | 0.00       |
| a  | b  | c  | 0.00       |

θ_{s|abe}
## Determinism

| A | B | C | Pr(S|A,B,E) |
|---|---|---|------------|
| a | b | c | 0.95       |
| a | b | c | 0.95       |
| a | b | c | 0.20       |
| a | b | c | 0.05       |
| ~a| b | c | 0.00       |
| ~a| b | c | 0.00       |
| ~a| b | c | 0.00       |
| ~a| b | c | 0.00       |

Tabular CPT

$$\lambda_{\sim a} \land \lambda_{b} \land \lambda_{c} \land \lambda_{s} \leftrightarrow \Theta_{s|\sim abc}$$

$$\sim \lambda_{\sim a} \lor \sim \lambda_{b} \lor \sim \lambda_{c} \lor \sim \lambda_{s}$$
Context-Specific Independence

| A  | B  | C  | Pr(S|A,B,C) |
|----|----|----|------------|
| a  | b  | c  | 0.95       |
| a  | b  | c | 0.95       |
| a  | b  | c | 0.20       |
| a  | b  | c | 0.05       |
| a  | b  | c | 0.00       |
| a  | b  | c | 0.00       |
| a  | b  | c | 0.00       |
| a  | b  | c | 0.00       |

\[ \lambda_a \land \lambda_b \land \lambda_c \land \lambda_s \leftrightarrow \Theta_{s|abc} \]

\[ \lambda_a \land \lambda_b \land \lambda_{\neg c} \land \lambda_s \leftrightarrow \Theta_{s|ab\neg c} \]

\[ \lambda_a \land \lambda_b \land \lambda_s \leftrightarrow \Theta_{s|ab} \]

Tabular CPT

A. Darwiche
The Ace System: 
http://reasoning.cs.ucla.edu/ace

Belief network

Arithmetic Circuit

CNF

Smooth d-DNNF

\[ \lambda_x \lor \lambda_{\neg x} \]

\[ \lambda_x \rightarrow \lambda_{\neg x} \]

\[ \lambda_x \land \lambda_y \rightarrow \theta_{x\land y} \]

\[ \lambda_x \land \lambda_y \rightarrow \theta_{x\land y} \]

\[ \lambda_x \ldots \lambda_y \ldots \theta_{x\land y} \ldots 1 \]

\[ \lambda_x \ldots \lambda_y \ldots \theta_{x\land y} \ldots \neg \lambda_x \ldots \]

A. Darwiche
1st International Evaluation of Exact Probabilistic Reasoning Systems (UAI’06, Boston)

- 135 problem instances from speech, coding, bioinformatics, circuits, medical diagnosis, ...
- Each team given 4 days of computation time
- UCLA: Only team to solve all problems in allotted time (solved all problems in 1 hr)
- Failure rates of other teams: 10%-40%

A. Darwiche
Inference by Compiling to d-DNNF

- **Deterministic Planning**
  - Blai Bonet and Hector Geffner (KR 2006)
- **Probabilistic Conformant Planning**
  - Jinbo Huang (ICAPS 2006)
- **Model-based diagnosis**
  - Paul Elliott and Brian Williams (AAAI 2006)
  - Anthony Barrett (IJCAI 2005)
- **Databases (query re-write)**
  - Yolife Arvelo, Blai Bonet and Maria Esther Vidal (AAAI 2006)
- **Inference in Bayesian Networks (2006 competition)**
  - Mark Chavira, Adnan Darwiche (IJCAI 2005)
- **Inference in Probabilistic Relational Models**
  - Mark Chavira, Adnan Darwiche and Manfred Jaeger (IJAR 2006)

---

c2d compiler:  [http://reasoning.cs.ucla.edu/c2d]