An Overview of Knowledge Compilation for Solving #SAT and its Applications to Cybersecurity

Presented by Richard Skowyra
What is \#SAT?

- SAT: NP-Complete Decision Problem
  - Is there an accepting path in a Nondeterministic Polytime TM?

- Is a WFF $\phi$ satisfiable?
  - Witness
  - Entailment
    - $\phi \models \alpha \iff Unsat(\phi \land \neg \alpha)$

- \#SAT: \#P-Complete Counting Problem
  - How many accepting paths?

- How many satisfying models does $\phi$ have?

- \#P is Hard: $PH \subseteq P^{\#P}$
Application: Model-Based Fault Diagnosis

- **System Model**
  - Boolean representation of device logic
  - Observables: I/O, Sensor Data, etc.
  - Assumables: Internal device states which must be inferred

- Smallest number of faults needed to produce observation?

- Minimum-cardinality diagnoses?

\[ \{A, \overline{B}, \overline{C}, Q\} \]

Diagram:
- System Model
- Observation
- Evaluator
- Diagnosis

Darwiche2001
Application: Probabilistic Inference

- Weighted model counting
  - Define theory $\Delta$ whose models are rows in joint distribution
  - Literal weights in model sum to probability of row in joint
  - Given evidence $e$:
    - Count $\Delta \land e$, summing over weights

- What is $Pr(e)$?
- What is most likely values of other variables?
- Can encode Bayesian networks

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$Pr$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>0.001</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$c_2$</td>
<td>0.002</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$c_3$</td>
<td>0.007</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$c_1$</td>
<td>0.009</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$c_2$</td>
<td>0.018</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$c_3$</td>
<td>0.063</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>0.0018</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$c_2$</td>
<td>0.0162</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$c_3$</td>
<td>0.162</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_2$</td>
<td>$c_1$</td>
<td>0.0072</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_2$</td>
<td>$c_2$</td>
<td>0.0648</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_2$</td>
<td>$c_3$</td>
<td>0.648</td>
</tr>
</tbody>
</table>

$e = \{a_1, c_1\}$

$Pr(e) = 0.001 + 0.009 = 0.01$

Darwiche2008
Application: Other NP-Complete Problems

- **Graph coloring**
  - How many?
    - When vertex $i$ is blue?
    - Where red is maximized?

- **Subset Sum**
  - How many?
    - Smallest/Largest subset?

- **Vertex Cover, Hamiltonian Path, etc.**
Application: Modeling Memory Security

Buffer Overflows
Return-into-Libc
Return-Oriented Programming

Data Execution Prevention
Address Space Layout Randomization
Information Leakage Attacks
• Compromise is possible if any attack is possible
• Attacks require \textit{Capabilities}
• Capabilities may rely on others
• Model: Attack Space is a Boolean Formula
  – Satisfiable iff an attack is possible
Modeling the Attack Space

- Compromise is possible if any attack is possible
- Attacks require *Capabilities*
- Capabilities may rely on others
- Model: Attack Space is a Boolean Formula
  - Satisfiable iff an attack is possible

\[ (ROP \rightarrow MLK \land GSN \land GA) \land \\
(MLK \rightarrow OfA \lor OnA) \land \\
(GSN \rightarrow \cdots) \land \\
\cdots \]

skowyra2013
Modeling Defenses

- Defenses must be enabled
  - Address Space Layout Randomization (ASLR)

- Defenses have *Weakness clauses* that allow bypass
  - ASLR can be bypassed with memory-disclosure attacks

- Defenses can disable a *Capability or Weakness*
  - ASLR disables Memory Layout Known

\[(D \land \neg W) \rightarrow \neg P\]
System Model

\[ \text{Attack} \land \left( \bigwedge \text{Defenses} \right) \]

- **SAT:**
  - Satisfaction: Is some attack ever possible?
  - Entailment: Is some attack always possible?

- **#SAT:**
  - How many attacks are possible?
    - When specific defenses are deployed?
  - ‘Best/Worst’ defense(s)?
  - What is the ‘simplest’ attack?
Direct #SAT Solving

• Exact Solutions:
  – DPLL-style approach extends standard SAT solvers
  – Exhaustive search, intractably expensive

• Approximate Solutions:
  – Stochastic search/sampling: Markov-Chain Monte Carlo, Simulated Annealing, etc.
  – Incomplete, few guarantees on approximation quality
Direct #SAT Solving

- **Exact Solutions:**
  - DPLL-style approach extends standard SAT solvers
  - Exhaustive search, intractably expensive

- **Approximate Solutions:**
  - Stochastic search/sampling: Markov-Chain Monte Carlo, Simulated Annealing, etc.
  - Incomplete, few guarantees on approximation quality

- **Knowledge Compilation**
  - Compile once to efficient data structure
  - Query frequently, in polytime
Knowledge Compilation

System Model

Compiler (Expensive)

Negation Normal Form*

Observations
Constraints
Min/Max Counts

Queries

Evaluator (Polynomial Time)

Offline

Online

Darwiche2001
- Naïve conversion to CNF can be exponential
- Tseitin Transformation guarantees linear increase in formula size by adding temporary variables
- Some Properties
  • $\text{Sat}(\phi) \leftrightarrow \text{Sat}(T(\phi))$ (Equisatisfiable)
  • $\phi \not\equiv T(\phi)$ (Semantic Inequivalence)
  • $\not\models T(\phi)$ (Invalidity)
Tseitin Transformation

\[ Q \land (x_0 \leftrightarrow A \land B) \land (x_1 \leftrightarrow B \lor C) \land (x_2 \leftrightarrow B \land C) \land (x_3 \leftrightarrow x_1 \land x_2) \land (Q \leftrightarrow x_0 \lor x_3) \]

Tseitin Transformation

**Biconditional**

\[ Q \land (x_0 \rightarrow A) \land (x_0 \rightarrow B) \land (x_0 \leftarrow A \land B) ... \]

**Material Implication**

\[ Q \land (\neg x_0 \lor A) \land (\neg x_0 \lor B) \land (x_0 \lor \neg A \lor \neg B) ... \]
• Exact Compilation: DNNF equivalent to CNF
  – $O(nw2^w)$ where $n = |\text{clauses}|$ and $w$ is dependent on an ordering of atoms

• Approximate Compilation: Three Options
  – Sound, but Incomplete: Gives up degree of decomposability
    – $\phi' \vdash \beta$ only if $\phi \vdash \beta$
  – Unsound, but Complete:
    • $\phi' \not\vdash \beta$ only if $\phi \not\vdash \beta$
  – Partial Decomposability: Sound and Complete except on certain atoms
    • Useful is set of observables is static and known

Darwiche2001
Negation Normal Form (NNF)

Rooted DAG Over Literals
Decomposability (DNNF)

- Polytime Operations:
  - Conditioning
  - Consistency
  - Clausal Entailment
  - Minimization
  - Model Enumeration
  - Projection
  - Satisfiability

∀α = α₁ ∧ … ∧ αₙ \mid atoms(α_i) \cap atoms(α_j) = \emptyset \text{ for } i \neq j

Darwiche2002
Determinism (d-DNNF)

- Additional Polytime Operations:
  - Counting
  - Validity

\[ \forall \alpha = \alpha_1 \lor \cdots \lor \alpha_n \mid \alpha_i \land \alpha_j \models False \text{ for } i \neq j \]
Tractable Operations on d-DNNF

- Satisfaction
- Clausal Entailment
- Conditioning
- Projection
- Minimum Cardinality
- Minimization
- Counting
- Enumeration

Linear Time
Satisfaction

\[ \text{Sat}(\Delta) \overset{\text{def}}{=} \begin{cases} \text{true}, & \text{if } \Delta \text{ is a literal or true;} \\ \text{false}, & \text{if } \Delta \text{ is false.} \end{cases} \]

\[ \text{Sat}(\Delta = \land_i \alpha_i) \overset{\text{def}}{=} \text{true iff Sat}(\alpha_i) \text{ is true for every } i. \]

\[ \text{Sat}(\Delta = \lor_i \alpha_i) \overset{\text{def}}{=} \text{true iff Sat}(\alpha_i) \text{ is true for some } i. \]

Darwiche2001
• $\Delta|\phi$ is the **conditioning** of $\Delta$ on $\phi$

• $\phi = \alpha_1 \land \cdots \land \alpha_n$

• Replace every $\Delta$-literal with TRUE or FALSE if it is consistent (inconsistent) with $\phi$

• Use: Incorporate observations

\[ \Delta|B \land C \]
• Does $\Delta$ entail clause $\beta$?

$Sat(\Delta|\beta') = False$ if $\Delta \models \beta$ where $\beta' \equiv \neg \beta$

$\beta = \neg A \lor \neg B \lor \neg C \lor \neg D$

$\beta' = A \land B \land C \land D$

$Sat(\Delta|\beta') = False$

$\Delta \models \beta$

Darwiche2001
Projection

- \( A \subseteq Atoms(\Delta) \)
- \( Project(\Delta, A) = \Gamma \)
  where:
  - \( \forall a \in \bar{A}, a = TRUE \)
  - \( \Gamma \) is an \( A \)-Sentence
  - For any \( A \)-Sentence \( \beta \),
    \( \Gamma \models \beta \) iff \( \Delta \models \beta \)
- Use: ‘Forget’ unimportant variables

\[ Project(\Delta, \{B, C\}) \]

Darwiche2001
Minimum Cardinity

\[
\text{MCard}(\Delta) = \begin{cases} 
0, & \text{if } \Delta \text{ is a positive literal or true}; \\
1, & \text{if } \Delta \text{ is a negative literal}; \\
\infty, & \text{if } \Delta \text{ is false}.
\end{cases}
\]

\[
\text{MCard}(\Delta = \lor_i \alpha_i) = \min_i \text{MCard}(\alpha_i).
\]

\[
\text{MCard}(\Delta = \land_i \alpha_i) = \sum_i \text{MCard}(\alpha_i).
\]

- Minimum number of FALSE variables
- Use: Minimal number of ‘faults’ to explain observation

Darwiche2001
Minimize($\Delta$) $\overset{\text{def}}{=} \Delta$, if $\Delta$ is a literal, true or false.

Minimize($\Delta = \lor_i \alpha_i$) $\overset{\text{def}}{=} \lor_{\text{MCard}(\alpha_i)=\text{MCard}(\Delta)} \text{Minimize}(\alpha_i)$.

Minimize($\Delta = \land_i \alpha_i$) $\overset{\text{def}}{=} \land_i \text{Minimize}(\alpha_i)$.

- **Minimize($\Delta$) = $\Gamma$ where:**
  - $\forall \omega, \omega \vDash \Gamma$ iff $\omega \vDash \Delta \land \text{Card}(\omega) = \text{MCard}(\Delta)$

- $\Gamma$ captures minimal solutions with respect to cardinality

- Use: Compute the set of simplest attacks, minimal diagnoses, etc.

Darwiche2001
Minimization

\[
\text{Minimize}(\Delta) \overset{\text{def}}{=} \Delta, \text{ if } \Delta \text{ is a literal, true or false.}
\]

\[
\text{Minimize}(\Delta = \bigvee_i \alpha_i) \overset{\text{def}}{=} \bigvee_{\text{MCard}(\alpha_i)=\text{MCard}(\Delta)} \text{Minimize}(\alpha_i).
\]

\[
\text{Minimize}(\Delta = \bigwedge_i \alpha_i) \overset{\text{def}}{=} \bigwedge_i \text{Minimize}(\alpha_i).
\]

- \(\text{Minimize}(\Delta) = \Gamma \) where:
  - \( \forall \omega, \omega \vDash \Gamma \text{ iff } \omega \vDash \Delta \land \text{Card}(\omega) = \text{MCard}(\Delta) \)
  
- \(\Gamma\) captures minimal solutions with respect to cardinality

- Use: Compute the set of simplest attacks, minimal diagnoses, etc.

Darwiche2001
Model Enumeration

\[ \text{models}(l) = \{\{l\}\} \] where \( l \) is a literal.

\[ \text{models}(\alpha_1 \lor \ldots \lor \alpha_n) = \text{models}(\alpha_1) \cup \ldots \cup \text{models}(\alpha_n). \]

\[ \text{models}(\alpha_1 \land \ldots \land \alpha_n) = \text{models}(\alpha_1) \times \ldots \times \text{models}(\alpha_n). \]

- \( \text{Models}(\Delta) = \{\omega | \omega \models \Delta\} \)
  - \( \Theta(mn) \) where:
    - \( m \) is the size of \( \Delta \) (number of gates)
    - \( n \) is \( |\text{Models}(\Delta)|^2 \)

- Polynomial in number of models
  - But potentially exponential number of models

Darwiche2001a
Model Enumeration

\[ \text{models}(l) = \{\{l\}\} \text{ where } l \text{ is a literal.} \]

\[ \text{models}(\alpha_1 \lor \ldots \lor \alpha_n) = \text{models}(\alpha_1) \cup \ldots \cup \text{models}(\alpha_n). \]

\[ \text{models}(\alpha_1 \land \ldots \land \alpha_n) = \text{models}(\alpha_1) \times \ldots \times \text{models}(\alpha_n). \]

Darwiche2001a
Model Enumeration

\[ \text{models}(l) = \{\{l\}\} \text{ where } l \text{ is a literal.} \]

\[ \text{models}(\alpha_1 \lor \ldots \lor \alpha_n) = \text{models}(\alpha_1) \cup \ldots \cup \text{models}(\alpha_n). \]

\[ \text{models}(\alpha_1 \land \ldots \land \alpha_n) = \text{models}(\alpha_1) \times \ldots \times \text{models}(\alpha_n). \]
Model Enumeration

$\text{models}(l) = \{\{l\}\}$ where $l$ is a literal.

$\text{models}(\alpha_1 \lor \ldots \lor \alpha_n) = \text{models}(\alpha_1) \cup \ldots \cup \text{models}(\alpha_n)$.

$\text{models}(\alpha_1 \land \ldots \land \alpha_n) = \text{models}(\alpha_1) \times \ldots \times \text{models}(\alpha_n)$.

Darwiche2001a
models(l) = {{l}} where l is a literal.

models(\(\alpha_1 \lor \ldots \lor \alpha_n\)) = models(\(\alpha_1\)) \cup \ldots \cup models(\(\alpha_n\)).

models(\(\alpha_1 \land \ldots \land \alpha_n\)) = models(\(\alpha_1\)) \times \ldots \times models(\(\alpha_n\)).
Model Enumeration

\[ \text{models}(l) = \{\{l\}\} \text{ where } l \text{ is a literal.} \]

\[ \text{models}(\alpha_1 \lor \ldots \lor \alpha_n) = \text{models}(\alpha_1) \cup \ldots \cup \text{models}(\alpha_n). \]

\[ \text{models}(\alpha_1 \land \ldots \land \alpha_n) = \text{models}(\alpha_1) \times \ldots \times \text{models}(\alpha_n). \]
Model Counting

\[ \text{val}(N) = 0 \text{ if } N \text{ is labeled with literal } l \text{ and } \neg l \notin S; \]
\[ \text{val}(N) = 1 \text{ if } N \text{ is labeled with literal } l \text{ and } \neg l \in S; \]
\[ \text{val}(N) = \prod_i \text{val}(N_i) \text{ if } N \text{ is labeled with } *, \text{ where } N_i \text{ are the children of } N; \]
\[ \text{val}(N) = \sum_i \text{val}(N_i) \text{ if } N \text{ is labeled with } +, \text{ where } N_i \text{ are the children of } N. \]

- S is a consistent set of literals
  - \( S = \emptyset \) if no constraints
- \( |\text{Models}(\Delta \cup S)| \) - Number of models given S
  - > 0 if \( \Delta \cup S \) is satisfiable

\[ S = A, \neg B \]
• Assertion: $|Models(\Delta \cup S \cup l)|$ for all $l \notin S$
  – What if $l$ is also constrained?

• Retraction: $|Models(\Delta \cup S \setminus \{l\})|$ for all $l \in S$
  – What if $l$ is no longer constrained?

• Flipping: $|Models(\Delta \cup S \setminus \{l\} \cup \{\neg l\})|$ for all $l \in S$
  – What if the constraint on $l$ is inverted?

• Use: What single change maximizes/minimizes the number of models?

• Can compute all three in linear time

Darwiche2001a
Counting Partial Derivatives

- G is linear in its variables (by decomposition)

**Assertion:** When \(-l \notin S\):

\[
\text{Models}^\#(\Delta \cup S \cup \{l\}) = \partial G(S)/\partial V_l.
\]

**Retraction:** When \(l \in S\):

\[
\text{Models}^\#(\Delta \cup S \setminus \{l\}) = \partial G(S)/\partial V_l + \partial G(S)/\partial V_{\neg l}.
\]

**Flipping:** When \(l \in S\):

\[
\text{Models}^\#(\Delta \cup S \setminus \{l\} \cup \{\neg l\}) = \partial G(S)/\partial V_{\neg l}.
\]

Darwiche2001a
Computing Partial Derivatives

\[ \text{PD}(N) = \begin{cases} 1, & \text{N is the root node;} \\ \sum_{M} \text{CPD}(M, N), & \text{otherwise;} \end{cases} \]

\[ \text{CPD}(M, N) = \begin{cases} \text{PD}(M), & \text{M is a + node;} \\ \text{PD}(M) \prod_{K \neq N} \text{VAL}(K), & \text{M is a \ast node;} \end{cases} \]

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

Assert \( \{C\} = \text{PD}(C) = 2 \)

Retract \( \{A\} \)

\( = \text{PD}(A) + \text{PD}(\neg A) = 1 \)

Flip \( \{\neg B\} = \text{PD}(B) = 2 \)
Lessons Learned in Implementation

• Good Domain Specific Languages are critical
  – Errors in the model will happen. Especially if:
    • Users need to write in Boolean logic
    • Namespaces overlap (or don’t) in unexpected ways
  – Need a basic type system (or at least sanity checks) to catch errors before expensive compilation phase

• Recursion is how algorithms are defined. Not necessarily how they should be implemented.
  – Tail-Recursion
  – While-loop ‘recursion’
Questions
References


• Darwiche2008: On probabilistic inference by weighted model counting. Artificial Intelligence 172(6), 772-779. 2008

• Skowyra2013: Systematic analysis of defenses against return-oriented programming. RAID 2013.