Logic in AI
Chapter 7
Mausam
(Based on slides of Dan Weld, Stuart Russell, Subbarao Kambhampati, Dieter Fox, Henry Kautz...)

Penguins are black and white. Some old TV shows are black and white. Therefore, some penguins are old TV shows.

Glasbergen

Logic: another thing that penguins aren’t very good at.

I am a nobody, and nobody is perfect; therefore I am perfect!

Owl Glen

02/03 CHAPPIE
Knowledge Representation

• represent knowledge about the world in a manner that facilitates inferencing (i.e. drawing conclusions) from knowledge.

• Example: Arithmetic logic
  – x >= 5

• In AI: typically based on
  – Logic
  – Probability
  – Logic and Probability
Common KR Languages

- **Prop logic**
  - First order predicate logic (FOPC)
  - First order Temporal logic (FOPC)
  - First order Prob. logic

- **Prob. Prop. logic**
  - Degree of belief
  - Degree of truth

- **Fuzzy Logic**
  - Degree of truth

- **Ontological commitment**
  - FOPC
  - Prob logic

- **Epistemological commitment**
  - Degree of belief
KR Languages

• Propositional Logic
• Predicate Calculus
• Frame Systems
• Rules with Certainty Factors
• Bayesian Belief Networks
• Influence Diagrams
• Ontologies
• Semantic Networks
• Concept Description Languages
• Non-monotonic Logic
Basic Idea of Logic

• By starting with true assumptions, you can deduce true conclusions.
Truth

• Francis Bacon (1561-1626)
No pleasure is comparable to the standing upon the vantage-ground of truth.

• Thomas Henry Huxley (1825-1895)
Irrationally held truths may be more harmful than reasoned errors.

• John Keats (1795-1821)
Beauty is truth, truth beauty; that is all ye know on earth, and all ye need to know.

• Blaise Pascal (1623-1662)
We know the truth, not only by the reason, but also by the heart.

• François Rabelais (c. 1490-1553)
Speak the truth and shame the Devil.

• Daniel Webster (1782-1852)
There is nothing so powerful as truth, and often nothing so strange.
Truth

• Francis Bacon (1561-1626)
  No pleasure is comparable to the standing upon the vantage-ground of truth.

• Thomas Henry Huxley (1825-1895)
  Irrationally held truths may be more harmful than reasoned errors.

• John Keats (1795-1821)
  Beauty is truth, truth beauty; that is all ye know on earth, and all ye need to know.

• Blaise Pascal (1623-1662)
  We know the truth, not only by the reason, but also by the heart.

• François Rabelais (c. 1490-1553)
  Speak the truth and shame the Devil.

• Daniel Webster (1782-1852)
  There is nothing so powerful as truth, and often nothing so strange.
Components of KR

• Syntax: defines the sentences in the language
• Semantics: defines the “meaning” to sentences
• Inference Procedure
  – Algorithm
  – Sound?
  – Complete?
  – Complexity
• Knowledge Base
Knowledge bases

- Knowledge base = set of sentences in a formal language

- **Declarative** approach to building an agent (or other system):
  - **Tell** it what it needs to know
  - Then it can **Ask** itself what to do - answers should follow from the KB

- Agents can be viewed at the **knowledge level**
  - i.e., what they know, regardless of how implemented

- Or at the **implementation level**
  - i.e., data structures in KB and algorithms that manipulate them
Propositional Logic

• Syntax
  – Atomic sentences: P, Q, ...
  – Connectives: $\wedge$, $\vee$, $\neg$, $\rightarrow$

• Semantics
  – Truth Tables

• Inference
  – Modus Ponens
  – Resolution
  – DPLL
  – GSAT
Propositional Logic: Syntax

• Atoms
  – P, Q, R, ...

• Literals
  – P, P

• Sentences
  – Any literal is a sentence
  – If S is a sentence
    • Then (S ∧ S) is a sentence
    • Then (S ∨ S) is a sentence

• Conveniences
  P → Q same as ¬P ∨ Q
Semantics

• *Syntax*: which arrangements of symbols are *legal*
  – (Def “sentences”)

• *Semantics*: what the symbols *mean* in the world
  – (Mapping between symbols and worlds)
Propositional Logic: SEMANTICS

• “Interpretation” (or “possible world”)
  – Assignment to each variable either T or F
  – Assignment of T or F to each connective via defns

<table>
<thead>
<tr>
<th></th>
<th>Q</th>
<th></th>
<th>P</th>
<th>Q</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>T</td>
<td></td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td></td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

P ∧ Q

<table>
<thead>
<tr>
<th></th>
<th>Q</th>
<th></th>
<th>P</th>
<th>Q</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>T</td>
<td></td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td></td>
<td>F</td>
<td></td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

P ∨ Q
Propositional Logic: SEMANTICS

- “Interpretation” (or “possible world”)
  - Assignment to each variable either T or F
  - Assignment of T or F to each connective via defns

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

P ∧ Q

P ∨ Q
Propositional Logic: **SEMANTICS**

- “Interpretation” (or “possible world”)
  - Assignment to each variable either T or F
  - Assignment of T or F to each connective via defns

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P</strong></td>
<td><strong>Q</strong></td>
<td><strong>P ∧ Q</strong></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P</strong></td>
<td><strong>Q</strong></td>
<td><strong>P ∨ Q</strong></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Satisfiability, Validity, & Entailment

• **S is satisfiable** if it is true in *some* world

• **S is unsatisfiable** if it is false *all* worlds

• **S is valid** if it is true in *all* worlds

• **S1 entails S2** if *wherever* S1 is true S2 is also true
Examples

\[ P \rightarrow Q \]
Examples

$P \rightarrow Q$

$R \rightarrow \neg R$
Examples

\[ P \rightarrow Q \]

\[ R \rightarrow \neg R \]

\[ S \land (W \land \neg S) \]
Examples

$P \rightarrow Q$

$R \rightarrow \neg R$

$S \land (W \land \neg S)$

$T \lor \neg T$
Examples

\( P \rightarrow Q \)

\( R \rightarrow \neg R \)

\( S \land (W \land \neg S) \)

\( T \lor \neg T \)

\( X \rightarrow X \)
Notation
Notation

\[ \implies \quad \not\implies \quad \rightarrow \quad \nrightarrow \quad \nrightarrow \]

} \text{Implication (syntactic symbol)}
Notation

Implication (syntactic symbol)

Proves: $S_1 \vdash_{ie} S_2$ if `ie' algorithm says `S2' from $S_1$

Entails: $S_1 \models S_2$ if wherever $S_1$ is true $S_2$ is also true
Notation

\[ \rightarrow \quad \rightarrow \]

\{ \}

Implication (syntactic symbol)

Proves: \( S_1 \vdash_{ie} S_2 \) if `ie' algorithm says `S2' from \( S_1 \)

Entails: \( S_1 \models S_2 \) if wherever \( S_1 \) is true \( S_2 \) is also true

- Sound
- Complete

- (all truth & nothing but the truth)
Notation

\[\equiv\]

\[\land\]

\[\rightarrow\]

\[\implies\]

\[\iff\]

Implication (syntactic symbol)

Proves: \(S_1 \models_{ie} S_2\) if `ie' algorithm says `S2' from \(S_1\)

Entails: \(S_1 \models S_2\) if wherever \(S_1\) is true \(S_2\) is also true

• Sound \(\models \rightarrow \models\)

• Complete \(\models \rightarrow \models\)

• (all truth & nothing but the truth)
Reasoning Tasks

• Model finding
  
  KB = background knowledge
  
  S = description of problem
  
  Show (KB \land S) is satisfiable
  
  A kind of constraint satisfaction

• Deduction
  
  S = question
  
  Prove that KB |= S
  
  Two approaches:
Reasoning Tasks

- **Model finding**
  - $KB = \text{background knowledge}$
  - $S = \text{description of problem}$
  - Show $(KB \land S)$ is satisfiable
  - A kind of *constraint satisfaction*

- **Deduction**
  - $S = \text{question}$
  - Prove that $KB \models S$
  - Two approaches:
    - **Rules to derive new formulas from old (inference)**
    - **Show $(KB \land \neg S)$ is unsatisfiable**
Special Syntactic Forms

• General Form:

\[ ((q \land \neg r) \rightarrow s) \land \neg(s \land t) \]

• Conjunction Normal Form (CNF)

\[ (\neg q \lor r \lor s) \land (\neg s \lor \neg t) \]

Set notation: \{ (\neg q, r, s), (\neg s, \neg t) \}

empty clause () = \textit{false}

• Binary clauses: 1 or 2 literals per clause

\[ (\neg q \lor r) \quad (\neg s \lor \neg t) \]

• Horn clauses: 0 or 1 positive literal per clause

\[ (\neg q \lor \neg r \lor s) \quad (\neg s \lor \neg t) \]

\[ (q \land r) \rightarrow s \quad (s \land t) \rightarrow \textit{false} \]
Propositional Logic: **Inference**

*A mechanical* process for computing new sentences

1. Backward & Forward Chaining
2. Resolution (Proof by Contradiction)
3. Davis Putnam
4. WalkSat
Inference 1: Forward Chaining

Forward Chaining
Based on rule of *modus ponens*

If know $P_1, \ldots, P_n$ & know $(P_1 \land \ldots \land P_n) \rightarrow Q$
Then can conclude $Q$

Backward Chaining: search

start from the query and go backwards
Analysis

• Sound?
• Complete?
Analysis

- Sound?
- Complete?

Can you prove

\[ \emptyset \models Q \lor \neg Q \]
Analysis

• Sound?
• Complete?

Can you prove
\[
\{ \} \models Q \lor \neg Q
\]

• If KB has only Horn clauses & query is a single literal
  – Forward Chaining is complete
  – Runs linear in the size of the KB
Example

$P \Rightarrow Q$
$L \land M \Rightarrow P$
$B \land L \Rightarrow M$
$A \land P \Rightarrow L$
$A \land B \Rightarrow L$
$A$
$B$
Example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Example

\[ P \Rightarrow Q \]
\[ L \wedge M \Rightarrow P \]
\[ B \wedge L \Rightarrow M \]
\[ A \wedge P \Rightarrow L \]
\[ A \wedge B \Rightarrow L \]
\[ A \]
\[ B \]
Example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \quad B \]
Example

- $P \Rightarrow Q$
- $L \land M \Rightarrow P$
- $B \land L \Rightarrow M$
- $A \land P \Rightarrow L$
- $A \land B \Rightarrow L$
- $A$
- $B$
Example

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]
Example

- $P \Rightarrow Q$
- $L \land M \Rightarrow P$
- $B \land L \Rightarrow M$
- $A \land P \Rightarrow L$
- $A \land B \Rightarrow L$
- $A$
- $B$
Example

$P \Rightarrow Q$
$L \land M \Rightarrow P$
$B \land L \Rightarrow M$
$A \land P \Rightarrow L$
$A \land B \Rightarrow L$
$A$
$B$
Propositional Logic: **Inference**

*A mechanical* process for computing new sentences

1. Backward & Forward Chaining
2. Resolution (Proof by Contradiction)
3. GSAT
4. Davis Putnam
Conversion to CNF

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]

1. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \( (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha) \).

\[ (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]

2. Eliminate \( \Rightarrow \), replacing \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \).

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

3. Move \( \neg \) inwards using de Morgan’s rules and double-negation:

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1}) \]

4. Apply distributivity law (\( \lor \) over \( \land \)) and flatten:

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]
Inference 2: Resolution
[Robinson 1965]

\{ (p \lor \alpha), (\neg p \lor \beta \lor \gamma) \} \vdash_{R} (\alpha \lor \beta \lor \gamma)
Inference 2: Resolution

[Robinson 1965]

\{ (p \lor \alpha), (\neg p \lor \beta \lor \gamma) \} \vdash_R (\alpha \lor \beta \lor \gamma)

Correctness

If S1 \vdash_R S2 then S1 \models S2

Refutation Completeness:

If S is unsatisfiable then S \vdash_R ()
Resolution subsumes Modus Ponens

A → B, A |= B
Resolution subsumes Modus Ponens

\[ A \rightarrow B, \ A \models B \]

\[ (\neg A \lor B) \]
Resolution subsumes Modus Ponens

\[ A \rightarrow B, \ A \models B \]

\[ (\neg A \lor B) \] \hspace{1cm} \text{(A)}
Resolution subsumes Modus Ponens

\[ A \rightarrow B, A \models B \]

\[
\begin{array}{c}
(\neg A \lor B) \\
(A) \\
(B)
\end{array}
\]
If Will goes, Jane will go
\[ \neg W \lor V \lor J \]
If doesn't go, Jane will still go
\[ W \lor V \lor J \]
Will Jane go?
\[ \models J \]
If Will goes, Jane will go
\(\sim W \lor J\)
If doesn't go, Jane will still go
\(W \lor J\)
Will Jane go?
\(\models J?\)
If Will goes, Jane will go
\[ \neg W \lor J \]
If doesn't go, Jane will still go
\[ W \lor J \]
Will Jane go?
\[ \models J \]
Resolution

*If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.*

Prove: *the unicorn is horned.*
Resolution

If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.

Prove: the unicorn is horned.

\[ M = \text{mythical} \]
\[ I = \text{immortal} \]
\[ A = \text{mammal} \]
\[ H = \text{horned} \]
Resolution

If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.

Prove: the unicorn is horned.

\[(\neg A \lor H) \quad (\neg I \lor H)\]

\[M = \text{mythical} \quad (M \lor A) \quad (\neg M \lor I)\]

\[I = \text{immortal}\]

\[A = \text{mammal}\]

\[H = \text{horned}\]
Resolution

If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.

Prove: the unicorn is horned.

\[
(\neg \ A \lor \ H) \quad (\neg \ H) \quad (\neg \ I \lor \ H) \\
\]

\[
M = \text{mythical} \quad (M \lor A) \quad (\neg M \lor I) \\
I = \text{immortal} \quad H = \text{horned} \\
A = \text{mammal} \]
Resolution

If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.

Prove: the unicorn is horned.

\[ (\neg A \lor H) \quad (\neg H) \quad (\neg I \lor H) \]

\[ (M \lor A) \quad (\neg A) \quad (\neg I) \quad (\neg M \lor I) \]

\( M = \) mythical
\( I = \) immortal
\( A = \) mammal
\( H = \) horned
Resolution

If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.

Prove: the unicorn is horned.

\[\neg A \lor H \quad \neg H \quad \neg I \lor H\]

\[M \lor A \quad \neg A \quad \neg I \quad \neg M \lor I\]

\[M \quad \neg M\]

\(M = \text{mythical}\)

\(I = \text{immortal}\)

\(A = \text{mammal}\)

\(H = \text{horned}\)
Resolution

If the unicorn is mythical, then it is immortal, but if it is not mythical, it is a mammal. If the unicorn is either immortal or a mammal, then it is horned.

Prove: the unicorn is horned.

\[ (\neg A \lor H) (\neg H) (\neg I \lor H) \]

\[ (M \lor A) (\neg A) (\neg I) (\neg M \lor I) \]

\[ (M) (\neg M) () \]

M = mythical
I = immortal
A = mammal
H = horned
Search in Resolution

- Convert the database into clausal form $D_c$
- Negate the goal first, and then convert it into clausal form $D_G$
- Let $D = D_c + D_G$
- Loop
  - Select a pair of Clauses $C_1$ and $C_2$ from $D$
    - Different control strategies can be used to select $C_1$ and $C_2$ to reduce number of resolutions tries
  - Resolve $C_1$ and $C_2$ to get $C_{12}$
  - If $C_{12}$ is empty clause, QED!! Return Success (We proved the theorem; )
  - $D = D + C_{12}$
- Out of loop but no empty clause. Return “Failure”
  - Finiteness is guaranteed if we make sure that:
    - we never resolve the same pair of clauses more than once;
    - we use factoring, which removes multiple copies of literals from a clause (e.g. QVPVP => QVP)

Idea 1: Set of Support: At least one of $C_1$ or $C_2$ must be either the goal clause or a clause derived by doing resolutions on the goal clause (*COMPLETE*)

Idea 2: Linear input form: Atleas one of $C_1$ or $C_2$ must be one of the clauses in the input KB (*INCOMPLETE*)
Model Finding

• Find assignments to variables that makes a formula true

• a CSP
Inference 3: Model Enumeration

for (m in truth assignments) {
    if (m makes $\Phi$ true) then return "Sat!"
}

return "Unsat!"
Inference 4: DPLL
(Enumeration of Partial Models)
[Davis, Putnam, Loveland & Logemann 1962]

Version 1

dpll_1(pa) {
    if (pa makes F false) return false;
    if (pa makes F true) return true;
    choose P in F;
    if (dpll_1(pa ∪ {P=0})) return true;
    return dpll_1(pa ∪ {P=1});
}

Returns true if F is satisfiable, false otherwise
DPLL Version 1

\[(a \lor b \lor c)\]
\[(a \lor \neg b)\]
\[(a \lor \neg c)\]
\[(\neg a \lor c)\]
DPLL Version 1

\((a \lor b \lor c)\)

\((a \lor \neg b)\)

\((a \lor \neg c)\)

\((\neg a \lor c)\)
DPLL Version 1

\[(F \lor b \lor c)\]
\[(F \lor \neg b)\]
\[(F \lor \neg c)\]
\[(T \lor c)\]
(F ∨ F ∨ c)
(F ∨ T)
(F ∨ ¬c)
(T ∨ c)
(F ∨ F ∨ F)
(F ∨ T)
(F ∨ T)
(T ∨ F)
DPLL Version 1
DPLL Version 1

\[(a \lor b \lor c)\]
\[(a \lor \neg b)\]
\[(a \lor \neg c)\]
\[(\neg a \lor c)\]
(a ∨ b ∨ c)
(a ∨ ¬b)
(a ∨ ¬c)
(¬a ∨ c)
DPLL as Search

• Search Space?

• Algorithm?
Improving DPLL

If literal $L_1$ is true, then clause $(L_1 \lor L_2 \lor ...)$ is true.

If clause $C_1$ is true, then $C_1 \land C_2 \land C_3 \land ...$ has the same value as $C_2 \land C_3 \land ...$.

Therefore: Okay to delete clauses containing true literals!

If literal $L_1$ is false, then clause $(L_1 \lor L_2 \lor L_3 \lor ...) \lor L_3 \lor ...)$ has the same value as $(L_2 \lor L_3 \lor ...)$.

Therefore: Okay to shorten clauses containing false literals!

If literal $L_1$ is false, then clause $(L_1)$ is false.

Therefore: the empty clause means false!
DPLL version 2

dpll_2(F, literal) {

    choose V in F;
    if (dpll_2(F, ¬V)) return true;
    return dpll_2(F, V);
}
dplll_2(F, literal) {
    remove clauses containing literal
    if (F contains no clauses) return true;
    shorten clauses containing \(-\text{literal}\)
    if (F contains empty clause) 
        return false;
    choose V in F;
    if (dplll_2(F, \(-V\))) return true;
    return dplll_2(F, V);
}
(F ∨ b ∨ c)
(F ∨ ¬b)
(F ∨ ¬c)
(T ∨ c)
DPLL Version 2

\[(b \lor c)\]

\[\neg b\]

\[\neg c\]
DPLL Version 2

\[(F \lor c)\]

\[(T)\]

\[(\neg c)\]
DPLL Version 2

(c)

(¬c)
DPLL Version 2

(F)

(T)
DPLL Version 2

Empty clause!

()
Structure in Clauses

• **Unit Literals**
  
  A literal that appears in a singleton clause
  
  \{\{\neg b \ c\}\{\neg c\}\{a \ \neg b \ e\}\{d \ b\}\{e \ a \ \neg c\}\}
Structure in Clauses

• Unit Literals

A literal that appears in a singleton clause

\{\neg b \ \neg c \ \neg c\} \{a \neg b \ e\} \{d \ b\} \{e \ a \neg c\}

*Might as well set it true! And simplify*

\{\neg b\} \ {a \neg b \ e}\{d \ b\}
Structure in Clauses

• Unit Literals

A literal that appears in a singleton clause

\{\{\lnot b \lor c\}\{\lnot c\}\{a \lor \lnot b \lor e\}\{d \lor b\}\{e \lor a \lor \lnot c\}\}

_Might as well set it true! And simplify_

\{\{\lnot b\}\{a \lor \lnot b \lor e\}\{d \lor b\}\}

\{\{d\}\}
Structure in Clauses

• Unit Literals

   A literal that appears in a singleton clause
   \{\neg b \land c\}{\neg c}{a \land \neg b \land e}{d \land b}{e \land a \land \neg c}\}

   *Might as well set it true! And simplify*

   \{\neg b\}{a \land \neg b \land e}{d \land b}\}

   \{d\}

• Pure Literals

   – A symbol that always appears with same sign

   – \{a \land \neg b \land c\}{\neg c \land d \land \neg e}{\neg a \land \neg b \land e}{d \land b}{e \land a \land \neg c}\}


Structure in Clauses

• Unit Literals

A literal that appears in a singleton clause

\[\{\neg b \; c\}\{\neg c\}\{a \; \neg b \; e\}\{d \; b\}\{e \; a \; \neg c\}\]

*Might as well set it true! And simplify*

\[\{\neg b\}\quad \{a \; \neg b \; e\}\{d \; b\}\]

\[\{\{d\}\}\]

• Pure Literals

– A symbol that always appears with same sign

– \[\{a \; \neg b \; c\}\{\neg c \; d \; \neg e\}\{\neg a \; \neg b \; e\}\{d \; b\}\{e \; a \; \neg c\}\]

*Might as well set it true! And simplify*

\[\{a \; \neg b \; c\} \quad \{\neg a \; \neg b \; e\} \quad \{e \; a \; \neg c\}\]
In Other Words

Formula \((L) \land C_2 \land C_3 \land \ldots\) is only true when literal \(L\) is true

Therefore: Branch immediately on unit literals!

May view this as adding constraint propagation techniques into play
In Other Words

Formula \((L) \land C_2 \land C_3 \land \ldots\) is only true when literal \(L\) is true.
Therefore: Branch immediately on unit literals!
If literal \(L\) does not appear negated in formula \(F\), then setting \(L\) true preserves satisfiability of \(F\).
Therefore: Branch immediately on pure literals!

May view this as adding constraint propagation techniques into play.
DPLL (previous version)
Davis – Putnam – Loveland – Logemann

dpll(F, literal) {
    remove clauses containing literal
    if (F contains no clauses) return true;
    shorten clauses containing \lnot literal
    if (F contains empty clause)
        return false;

    choose V in F;
    if (dpll(F, \lnot V)) return true;
    return dpll(F, V);
}
DPLL (for real!)
Davis – Putnam – Loveland – Logemann

dplll(F, literal) {
    remove clauses containing literal
    if (F contains no clauses) return true;
    shorten clauses containing \neg\text{literal}
    if (F contains empty clause)
        return false;
    if (F contains a unit or pure L)
        return dplll(F, L);
    choose V in F;
    if (dplll(F, \neg V)) return true;
    return dplll(F, V);
}
DPLL (for real)

\[(a \lor b \lor c)\]
\[(a \lor \neg b)\]
\[(a \lor \neg c)\]
\[(\neg a \lor c)\]
DPLL (for real!)
Davis – Putnam – Loveland – Logemann

dpll(F, literal){
    remove clauses containing literal
    if (F contains no clauses) return true;
    shorten clauses containing \( \neg \)literal
    if (F contains empty clause) return false;
    if (F contains a unit or pure L) return dpll(F, L);
    choose V in F;
    if (dpll(F, \( \neg V \)) return true;
    return dpll(F, V);
}
Heuristic Search in DPLL

- Heuristics are used in DPLL to select a (non-unit, non-pure) proposition for branching
- Idea: identify a most constrained variable
Heuristic Search in DPLL

• Heuristics are used in DPLL to select a (non-unit, non-pure) proposition for branching

• Idea: identify a most constrained variable
  – Likely to create many unit clauses
Heuristic Search in DPLL

• Heuristics are used in DPLL to select a (non-unit, non-pure) proposition for branching

• Idea: identify a most constrained variable
  – Likely to create many unit clauses

• MOM’s heuristic:
  – Most occurrences in clauses of minimum length
Success of DPLL

• 1962 – DPLL invented
• 1992 – 300 propositions
• 1997 – 600 propositions (satz)
• Additional techniques:
  – Learning conflict clauses at backtrack points
  – Randomized restarts
  – 2002 (zChaff) 1,000,000 propositions – encodings of hardware verification problems