Overview

⇒ Integrity Constraints
• Disjunctive & Negative Knowledge
• Resolution Rule
• Bottom-Up
• Proof by Refutation (Horne)
• Top-Down (Disj. and Neg. Knowledge)
Integrity Constraints (Chapter 7.3)

• **Integrity Constraint**
  - \( \text{false} \leftarrow a_1 \land \ldots \land a_n \)
  - Means that \( a_1 \land \ldots \land a_n \) cannot be true
    + A model of KB must make each clause true
      If it makes the body true, then ‘false’ must be true
      Which is a contradiction, so body can’t be true
  - Allows us to specify things that should not be true
  - \( \text{false} \leftarrow a \) means that \( a \) has to be false in all models

• **Horne Clause**
  - Is either a clause or an integrity constraint
  - Have either an atom or a false on left hand side
Example

- KB
  
  \( \text{false} \leftarrow a \land b \)
  
  \( a \leftarrow c \)
  
  \( b \leftarrow c \)
  
  - Can conclude that \( c \) is false in all models of KB
  
  - If interpretation makes \( c \) true,
    it would also make \( a \) and \( b \),
    hence would make \( a \land b \) true
    and hence would need to be make \( \text{false} \) true contradiction
  
  - So \( KB \models \neg c \)
Syntax and Semantics of Not

• Add \( \neg \) to syntax

• Semantics
  - If interpretation I and variable assignment \( \rho \) make \( a \) true, than it makes \( \neg a \) false, and vice versa
  - \( KB \models \neg c \)
    + If \( \neg c \) is true in all models (and with all variable assignments)
    + If \( c \) is false in all models

• For any \( d \)
  - \( KB \models d \) if \( d \) is true in all models of \( KB \)
  - \( KB \models \neg d \) if \( d \) is false in all models of \( KB \)
  - otherwise
Variations on Integrity Constraints

• $false \leftarrow a$
  - Can be written as $\neg a$
  - Or as $\neg a \leftarrow true$

• $false \leftarrow a_1 \land \ldots \land a_n$
  - Can be written as $\neg a_1 \leftarrow a_2 \land \ldots \land a_n$
  - Can be written as $\neg a_1 \lor \ldots \lor \neg a_n$

• $h \leftarrow a_1 \land \ldots \land a_n$
  - Can be written as $h \lor \neg a_1 \lor \ldots \lor \neg a_n$: conjunctive normal form
    + a conjunction (KB) of disjunctions (each clause in KB)
  - Negative signs just on atoms
    + Call atom with optional negation a literal

• Horne clause
  - When written in conjunctive normal form, at most one positive literal
Syntax and Semantics of Or

• Add $\lor$ to syntax

• Semantics
  - If interpretation $I$ and variable assignment $\rho$ make $a$ true or $b$ true, then it makes $a \lor b$ true
Unsatisfiable

- Integrity constraints means there might not be a model of a KB
  - KB is *unsatisfiable*
  - Proof procedure should be able to derive false
  - Proof procedure should derive false iff KB is unsatisfiable

- Example KB
  
  \[
  a \\
  false \leftarrow a
  \]
  - Unsatisfiable
Horne Clauses

• Integrity constraints can be used for diagnostics
  - See textbook

• Also allows us a way to state some negative information
  - From electrical domain: a light can’t be both on and off
  - Should be able to prove negative facts as well

• Our top-down and bottom-up proof procedures not powerful enough
Overview

• Integrity Constraints

⇒ Disjunctive & Negative Knowledge

• Resolution Rule

• Bottom-Up

• Proof by Refutation (Horne)

• Top-Down (Disj. and Neg. Knowledge)
Disjunctive Knowledge

<table>
<thead>
<tr>
<th></th>
<th>In Conjunctive Normal Form</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive literals</td>
</tr>
<tr>
<td>Datalog</td>
<td>Exactly 1</td>
</tr>
<tr>
<td>Horne</td>
<td>At most 1</td>
</tr>
<tr>
<td>Disjunctive &amp; Negative</td>
<td>any number</td>
</tr>
</tbody>
</table>

- Remove restriction of Horne clauses
  - Allow disjunction in head
  - \( c \lor b \leftarrow a \)
  - Allow atoms to be negated
  - Can be converted into conjunctive normal form
    + No restrictions on how many positive literals
- In fact allow any combination with \( \land, \lor, \neg, \) and \( \leftarrow \)
Conversion to Conjunctive Normal Form

• Any expression with $\land$, $\lor$, $\neg$, and $\leftarrow$ can be converted into a set of clauses in conjunctive normal form
  - no literals on right hand side of $\leftarrow$

• Step 1: Eliminate $\leftarrow$
  - $(\phi \leftarrow \psi)$ replace with $(\phi \lor \neg \psi)$

• Step 2: Distribute negationation so only applies to atoms
  - $\neg \neg \phi$ replaced with $\phi$
  - $\neg (\phi \lor \psi)$ replaced with $\neg \phi \land \neg \psi$
  - $\neg (\phi \land \psi)$ replaced with $\neg \phi \lor \neg \psi$

• Step 3: Distribute $\lor$’s over $\land$’s
  - $(\phi \lor (\psi \land \chi))$ replaced by $((\phi \lor \psi) \land (\phi \lor \chi))$
Example

\( a \land (b \lor c \lor \neg (d \leftarrow e)) \)
\( a \land (b \lor c \lor \neg (d \lor e)) \)
\( a \land (b \lor c \lor \neg d \land \neg e) \)
\( a \land (b \lor c \lor \neg d \land e) \)
\( a \land (b \lor (c \lor \neg d \land e)) \)
\( a \land (b \lor (c \lor \neg d \land e)) \)
\( a \land (b \lor ((c \lor \neg d) \land (c \lor e))) \)
\( a \land ((b \lor (c \lor \neg d)) \land (b \lor (c \lor e))) \)
\( a \land (b \lor c \lor \neg d) \land (b \lor c \lor e) \)

• Result: 3 clauses
  
  \( a \)
  
  \( b \lor c \lor \neg d \)
  
  \( b \lor c \lor e \)
Conjunctive Normal Form

- Convert all clause into conjunctive normal form
  - Can view the literals as a set
    - Duplicates are removed
    - $a \leftarrow b \land c \land c$
    - $a \lor \neg b \lor \neg c \lor \neg c$
    - $\{a, \neg b, \neg c\}$

- Our KB is now a set of clauses, where each clause is a set of literals
• Recap: $a \lor (b \land c)$
  
  $(a \lor b) \land (a \lor c)$

Knowledge base: \{a \lor b, a \lor c\}

Or in set notation for disjuncts: \{\{a, b\}, \{a, c\}\}

• Now with variables:

  $\text{male}(X) \lor (\text{female}(X) \land \text{ownsdog}(X))$
  
  $(\text{male}(X) \lor \text{female}(X)) \land (\text{male}(X) \lor \text{ownsdog}(X))$

  - If we show the implicit quantifier
    
    $\forall X (\text{male}(X) \lor \text{female}(X)) \land (\text{male}(X) \lor \text{ownsdog}(X))$

  - Variables in two disjuncts seem bound together
    
    + Can we write it as two separate clauses, each with own $X$?

  - Yes, means the same thing: $\forall X \text{male}(X) \lor \text{female}(X)$
    
    $\forall X \text{male}(X) \lor \text{ownsdog}(X)$

  - So, just as in the non-variable case, we can write this as:
    
    $\{\{\text{male}(X), \text{female}(X)\}, \{\text{male}(X), \text{ownsdog}(X)\}\}$
Overview

• Integrity Constraints
• Disjunctive & Negative Knowledge

⇒ Resolution Rule

• Bottom-Up

• Proof by Refutation (Horne)

• Top-Down (Disj. and Neg. Knowledge)
Resolution Rule

- Resolution Rule
  - Resolvent A includes $\neg a$
  - Resolvent B includes $b$
  - $a$ and $b$ can be unified
  - $\sigma$ is the MGU of $a$ and $b$
  - Let $A'$ be $A$ with $\neg a$ removed, and $\sigma$ applied
  - Let $B'$ be $B$ with $b$ removed, and $\sigma$ applied
  - Resolvent is $A' \cup B'$
Examples

• \{a, \neg b, \neg c\} with \{d, \neg c, b\}

• \{a(X), b(X, Y), \neg c(Y)\} with \{\neg b(Z, Z), c(a)\}
Overview

• Integrity Constraints
• Disjunctive & Negative Knowledge
• Resolution Rule

⇒ Bottom-Up

• Proof by Refutation (Horne)
• Top-Down (Disj. and Neg. Knowledge)
Bottom Up Proof Procedure (Section 7.5)

• Find set of ‘minimal’ truths
• Set C to KB
• Repeat:
  + Pull two clauses from C
  + Apply resolution rule if you can, giving R
  + If R contains A and ¬A, skip R, since trivally true
  + If there is an \( R' \in C \) such that \( R' \subseteq R \), skip R, since already implied
  + If there are any \( R' \in C \) such that \( R \subseteq R' \), remove \( R' \) since now implied by R

• Consequent set no longer just has atoms in it, but can have any arbitrary clause in conjunctive normal form
  - But still just has minimal truths in it
Queries to Bottom-Up Proof Procedure

- Query can be disjunction of positive or negative literals
  - For previous bottom-up procedure, was just a single positive literal
- Write query as a set of literals
  - If query contains $A$ and $\neg A$ obviously true
    + Since one of them is true in any model, so disjunction true in all models
  - If there is a member $e$ of $C$ and a substitution $\sigma$
    such that $e\sigma$ is a subset of the query, then the query is true
Example 1

\[
\begin{align*}
\text{false} & \leftarrow a \land b \\
a & \leftarrow c \\
b & \leftarrow c \\
\neg c & \leftarrow \neg c
\end{align*}
\]
Example 1: Answer

- Convert KB to CNF

\[ \text{false} \leftarrow a \land b \quad \neg a \lor \neg b \quad 1 \]
\[ a \leftarrow c \quad a \lor \neg c \quad 2 \]
\[ b \leftarrow c \quad b \lor \neg c \quad 3 \]

- KB added to consequent set \( C \)

- Look for more clauses to add to \( C \)

\[ \neg b \lor \neg c \quad \text{use 1 \& 2} \quad 4 \]
\[ \neg a \lor \neg c \quad \text{use 1 \& 3} \quad 5 \]
\[ \neg c \quad \text{use 1 \& 5} \quad 6 \]

- \( \neg c \) allows us to prune out 2, 3, 4, 5!
- Nothing else can be done (need to keep going until nothing can be added)

- Final Consequent set \( C' \):

\[ \neg a \lor \neg b \quad 1 \]
\[ \neg c \quad 6 \]

- \( \neg c \) is in \( C' \), so query is true
Example II

\[(a \lor \neg b) \leftarrow c\]
\[\neg e \leftarrow \neg c\]
\[b \lor d\]
\[(a \lor b) \leftarrow d\]
\[e \leftarrow \neg a\]

\[a \lor c\]
Sound & Complete?

- Is it bottom-up strategy sound and complete?
Overview

• Integrity Constraints
• Disjunctive & Negative Knowledge
• Resolution Rule
• Bottom-Up

⇒ Proof by Refutation (Horne)

• Top-Down (Disj. and Neg. Knowledge)
Horne Clauses and Resolution (not in textbook)

• What is so special about Horne clauses?
  - More powerful than Datalog
  - **And** an efficient search solution for false

• If $KB \models \{\}$
  - Where $\{\}$ represents the empty clause, which is the same as false
    + No disjuncts in the clause means nothing can make cause true
  - Means $KB$ has no model, which means it is inconsistent

• Why is being able to prove a KB is inconsistent useful?
  - Say we want to prove $KB \models q$, where $q$ is a literal
  - What would happen if we add $\neg q$ into KB?
Inconsistencies

• $KB \models q$ is true if $KB \cup \{\neg q\} \models \text{false}$

• Given a $KB'$, there is an efficient way to see if it is inconsistent
Unit Resolution

• Pick two resolvents where one of them is a unit clause
  - This is a restricted bottom-up proof procedure

• Unit Resolution always halts
  - For Horn Clauses & no functions
  - Proof
    + Let the largest clause in KB have $k$ literals
    + Results always have fewer than $k$ literals
    + Finite number legal literals
    + Finite number of results that Unit resolution can find

• We say it is refutation complete
Example 1

- Convert to CNF

\[
\begin{array}{lll}
false & \leftarrow & a \land b \\
\neg a \lor \neg b & \leftarrow & 1 \\
a & \leftarrow & c \\
a \lor \neg c & \leftarrow & 2 \\
b & \leftarrow & c \\
b \lor \neg c & \leftarrow & 3 \\
\neg c & \leftarrow & ? \\
c & \leftarrow & 4 \\
\end{array}
\]
Overview

• Integrity Constraints
• Disjunctive & Negative Knowledge
• Resolution Rule
• Bottom-Up
• Proof by Refutation (Horne)

⇒ Top-Down (Disj. and Neg. Knowledge)
Top Down Proof Procedure

• Start with query, which is a conjunction of literals
  - \( \text{yes} \leftarrow p_1 \land \ldots \land p_i \land \neg p_{i+1} \land \ldots \land \neg p_n \)
  - For previous top-down procedure, it was conjunction of positive literals

• Turn into disjunctive normal form
  - \( \{\text{yes}, \neg p_1, \ldots, \neg p_i, p_{i+1}, \ldots, p_n\} \)

• Use resolution rule to derive new answer clauses
  - Attack first non-yes literal in answer clause
  - Stop when just “yes” atom in answer clause
A problem

• Consider KB
  \[ a \lor b \]
  \[ c \leftarrow a \]
  \[ c \leftarrow b \]
  into disjunctive normal form \[ \Rightarrow \]
  \[ c \lor \neg a \]
  \[ c \lor \neg b \]

• Proof
  \[ ?c \]
  yes \lor \neg c
  Use \[ c \lor \neg a \]
  yes \lor \neg a
  Use \[ a \lor b \]
  yes \lor b
  Use \[ c \lor \neg b \]
  yes \lor c
  Use \[ c \lor \neg b \]

- Now what?
Negative Ancestor Rule

• Can view proof as
  + adding original answer clause KB
  + and trying to prove \(yes\) by itself
• So, should be sound to resolve answer clause with a previous answer clause
  + Didn’t need to do this for Datalog
    as it did not need this to make proof procedure complete
  + But we do need this ability here

• Proof

\[
\begin{align*}
yes \vee \neg c & \quad \text{Use } c \vee \neg a \text{ from KB} \\
yes \vee \neg a & \quad \text{Use } a \vee b \text{ from KB} \\
yes \vee b & \quad \text{Use } c \vee \neg b \text{ from KB} \\
yes \vee c & \quad \text{Use } yes \vee \neg c \text{ from prior answer clause in proof} \\
yes & \\
\end{align*}
\]
Disjunctive Answers

- KB
  \[ p(X) \leftarrow q(X) \quad p(X) \lor \neg q(X) \]
  \[ q(a) \lor q(b) \quad q(a) \lor q(b) \]

- Query \( p(X) \)
  - yes, with either \( X = a \) or \( X = b \), but you don’t know which
  - Very different from answer that it has two solutions,
    one with \( X = a \) and another with \( X = b \)
    + Which you would have gotten if \( q(a) \) and \( q(b) \) was replaced with \( q(a) \land q(b) \)

- We would like proof procedure to find \( yes(a) \lor yes(b) \)
  - Use Top-down proof procedure
  - Delay literals of the form \( yes(X) \)
  - Stop when just \( yes \) atoms left in answer clause
Example

- KB
  \[ p(X) \leftarrow q(X) \]
  \[ q(a) \lor q(b) \lor q(c) \]
  \[ yes(X) \leftarrow p(X) \]
  into disjunctive normal form \[ \Rightarrow \]
  \[ p(X) \lor \neg q(X) \]
  \[ q(a) \lor q(b) \lor q(c) \]
  \[ yes(X) \lor \neg p(X) \]

- Proof
Answer Clause
\[\text{yes}(X) \lor \lnot p(X)\]
\[\text{yes}(X) \lor \lnot q(X)\]
\[\text{yes}(a) \lor q(b) \lor q(c)\]
\[\text{yes}(a) \lor p(b) \lor q(c)\]
\[\text{yes}(a) \lor \text{yes}(b) \lor q(c)\]
\[\text{yes}(a) \lor \text{yes}(b) \lor p(c)\]
\[\text{yes}(a) \lor \text{yes}(b) \lor \text{yes}(c)\]

Rule applied
\[p(X) \lor \lnot q(X)\]
\[q(a) \lor q(b) \lor q(c)\]
\[\text{yes}(X) \lor \lnot q(X)\]
\[p(X) \lor \lnot q(X)\]
\[\text{yes}(X) \lor \lnot p(X)\]
\[p(X) \lor \lnot q(X)\]
\[\text{yes}(X) \lor \lnot p(X)\]
\[\text{yes}(X) \lor \lnot p(X)\]