Example

**Example**

- **Integrity Constraint**
  - false ← \( a \land b \)
  - **Consequence**
    - false
    - \( c \rightarrow q \)
    - \( c \rightarrow a \land b \)
    - \( q \lor a \lor \neg c \)
    - Can conclude that \( c \) is false in all models of \( KB \)
    - If interpretation makes \( c \) true, it would also make \( a \) and \( b \) true, hence would make \( a \land b \) true and hence would need to be made false.
    - **Contradiction**
    - So \( KB \models \neg c \)

**Overview**

- **Top-Down (Disj. and Neg. Knowledge)**
- **Proof by Resolution (Horne)**
- **Resolution Rule**
- **Disjunctive & Negative Knowledge**
- **Integrity Constraints (Chapter 7.3)**
Syntax and Semantics of Or

Add ¬ to syntax

Semantics

If interpretation I and variable assignment U make a or b true, then it makes a ∨ b true.

Variations on Integrity Constraints

Add ¬ to syntax

Semantics

Horne clause

When written in conjunctive normal form, at most one positive literal.

 Conjunctive normal form

∀a¬∀b¬...¬∀c¬d→f

¬a ∧ ¬b ∧ ¬c ∧ d → f

¬a ∧ ¬b ∧ ¬c ∧ ¬d → false

¬a ∧ ¬b ∧ ¬c ∧ a → true

¬a ∧ ¬b ∧ ¬c ∧ ¬a → false

false → a

true → ¬a

Otherwise

P is false in all models of Y

Y ⊢ ¬p

For any p

If p is false in all models

¬p ∈ F

If p is false in all models and with all variable assignments

¬p ∈ F

¬p ∈ F

If interpretation I and variable assignment U make a true, then it makes ¬a false.

Add ¬ to syntax

Semantics
Overview

- Top-Down (Disj. and Neg. Knowledge)
- Proof by Resolution (Horne)
- Bottom-Up
- Resolution Rule

= Disjunctive & Negative Knowledge
= Integrity Constraints

Horne Clauses

- Integrity constraints can be used for diagnostics
- Our top-down and bottom-up proof procedures not powerful enough
- Should be able to prove negative facts as well
- From decidable domains a higher can be both on and off
- Also allows us a way to state some negative information
- See textbook

Example KB

\[ \text{false} \rightarrow a \]

Example KB

- Proof procedure should derive false if KB is unsatisfiable
- Proof procedure should be able to derive false
- KB is unsatisfiable
- Integrity constraints mean there must not be a model of a KB

Unsatisfiable
Example

\[(\lambda \phi) \lor (\delta \lor \phi)\]

\[(\lambda \phi) \lor (\delta \lor \phi)\] - Step 3: Distribute \lor over \land.

\[(\delta \lor \neg \phi)\] - Step 2: Distribute negation so only applies to atoms.

\[(\delta \lor \neg \phi)\] - Step 1: Eliminate \rightarrow.

\[\neg \phi \lor \neg (\delta \lor \psi) \equiv \neg \phi \land \neg \psi\] - Eliminate negated expressions in the head.

Any expression with \lor, \land, \neg, and \rightarrow can be converted into a set of clauses in conjunctive normal form.

Conversion to Conjunctive Normal Form

- Eliminate \rightarrow:
  - \(\phi \rightarrow \psi\) replaced with \((\phi \lor \neg \psi)\)

- Distribute negation:
  - \(\neg \neg \phi\) replaced with \(\phi\)
  - \(\neg (\phi \lor \psi)\) replaced with \(\neg \phi \land \neg \psi\)
  - \(\neg (\phi \land \psi)\) replaced with \(\neg \phi \lor \neg \psi\)

- Distribute \lor:
  - \((\phi \lor (\psi \land \chi))\) replaced by \(((\phi \lor \psi) \land (\phi \lor \chi))\)

- In fact allow any combination with \lor, \land, \neg, and \rightarrow

<table>
<thead>
<tr>
<th>Disjunctive Knowledge</th>
<th>Negated Herads</th>
<th>Positive Herads</th>
<th>At most 1</th>
<th>Exactly 1</th>
<th>any number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disjunctive &amp; Negated</td>
<td>Negated Herads</td>
<td>Positive Herads</td>
<td>any number</td>
<td>any number</td>
<td>any number</td>
</tr>
<tr>
<td>In Conjunctive Normal Form</td>
<td>Negated Herads</td>
<td>Positive Herads</td>
<td>any number</td>
<td>any number</td>
<td>any number</td>
</tr>
</tbody>
</table>

Datalog

- Exactly 1 any number

Horne

- Allow disjunction in head
- No restrictions on how many positive heads
- Can be converted into conjunctive normal form
- Allow atoms to be negated
- Can be converted into conjunctive normal form

- Removes restriction of Horn clauses

- In fact allow any combination with \lor, \land, \neg, and \rightarrow
Overview

- Integrity Constraints
- Disjunctive & Negative Knowledge
- Resolution Rule
- Bottom-Up
- Proof by Refutation (Horne)
- Top-Down (Proof of Negation (negli))

Variables

- Recap:
  \( a \lor (b \land c) \)
  \( (a \lor b) \land (a \lor c) \)

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

- 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 \( \forall X \)

- Variables in two disjuncts seem bound together

- Can we write it as two separate clauses, each with own \( X \)?

- Yes, means the same thing.

- 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) \} \} \}

- Where each clause is a set of literals

- Our KB is now a set of clauses.

- Convert all clauses into conjunctive normal form
Overview

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

Examples

\{ \{ p(x,y), q(x,y), \neg r \} \text{ with } \{ \{ \neg p(y,z), \neg q(z,y) \} \} \}

\{ \{ \neg p(x,y), \neg q(x,y), \neg r \} \text{ with } \{ \{ p(x,y) \} \} \}

Resolution Rule

• Resolution Rule
  - Resolvent A includes \( \neg a \)
  - Resolvent B includes \( b \)
  - \( a \) and \( b \) can be unified
  - \( \sigma \) is the Most General Unifier 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 \( A' \cup B' \)
Example I

\[
\begin{align*}
\text{false} & \leftarrow a \wedge b \\
a & \leftarrow c \\
b & \leftarrow c \\
n & \rightarrow a \wedge b
\end{align*}
\]

Queries to Bottom-Up Proof Procedure

- A query can be a disjunction of positive or negative literals.
  - For the previous bottom-up procedure, was just a single positive literal.
- Write a query as a set of literals.
  - If a query contains \( A \) and \( \neg A \), obviously true.
  - Since one of them is true in any model.
  - If \( A \) and \( \neg A \) are in a query, then \( A \) or \( \neg A \) is obviously true.
- Write the goal as a set of literals.
- Queries can be disjunction of positive or negative literals.

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.
    - If there is a \( A \) and \( \neg A \) in \( C \), skip.
    - Since one of them is true in any model.
    - If \( A \) and \( \neg A \) are in a query, then \( A \) or \( \neg A \) is obviously true.
  - If there is an \( R' \in C \) such that \( R \subset R' \), skip.
    - Since one of them is true in any model.
  - If there is an \( R' \in C \) such that \( R \subset R' \), remove.
    - Since now implied.
- Consequent set no longer just has atoms in it.
  - But can have any arbitrary clause in conjunctive normal form.
  - If there is any \( \neg A \) such that \( C \cup \neg A \) contains \( R \), remove \( R \).
  - Since now implied.
  - If \( C \cup \neg A \) contains \( R \), then \( R \) can be extended.
  - Apply resolution rule if you can.
  - Pull two clauses from \( C \).
- Repeat.

Stop if \( C \subseteq KB \).

Find set of minimal truths.
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 $\text{KB} | \emptyset = \emptyset$
- Where $\emptyset$ represents the empty clause, which is the same as false
- No disjuncts in the clause means nothing can make clause true
- $\emptyset$ represents the empty clause, which is the same as false
- If $\text{KB} | \emptyset = \emptyset$
- And an efficient search solution for false
- More powerful than Datalog
- Why is being able to prove a $\text{KB}$ is inconsistent useful?
- Say we want to prove $\text{KB} | q = \emptyset$, where $q$ is a literal
- What would happen if we add $\neg q$ into $\text{KB}$?
- Why is being able to prove a $\text{KB}$ is inconsistent useful?
- Say we want to prove $\text{KB} | q = \emptyset$, where $q$ is a literal
Overview

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

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 complete
- Given a KB, there is an efficient way to see if it is inconsistent
  \[ \{ \text{false} \} \cap \{ \text{false} \} \]

Inconsistencies
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
  + 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:

\[ \text{Proof } \]
\[ \text{CSE560 Class 10: 29 c } \]

A problem

- Consider KB
  \[ a \lor b \]
  \[ a \lor b \]
  \[ c \leftarrow a \]
  \[ c \leftarrow b \]
- Now what?
  \[ \text{KB} \]

Top Down Proof Procedure

- Start with query, which is a conjunction of literals
  \[ \text{yes} \lor \neg \text{p}_1 \land \ldots \land \neg \text{p}_i \land \text{p}_{i+1} \land \ldots \land \neg \text{p}_n \]
- For previous top-down procedure, it was conjunction of positive literals
- Turn into disjunctive normal form
- Use resolution rule to derive new answer clauses
- Attack first non-yes literal in answer clause
- Stop when just "yes" atom in answer clause

Proof:

\[ \text{Proof } \]
\[ \text{CSE560 Class 10: 28 c } \]
Example

\[ (X)d \land (X)e \land (a) \]
\[ (X)b \land (q) \land (u) \land (b) \]
\[ (X)\neg (X)d \]
\[ KB \]

Proof

\[ (X)d \land (X)e \land (a) \]
\[ (X)b \land (q) \land (u) \land (b) \]
\[ (X)\neg (X)d \]
\[ KB \]

Risk affected

\[ (X)d \land (X)e \land (a) \]
\[ (X)b \land (q) \land (u) \land (b) \]
\[ (X)\neg (X)d \]
\[ KB \]

Example

\[ (X)d \land (X)e \land (a) \]
\[ (X)b \land (q) \land (u) \land (b) \]
\[ (X)\neg (X)d \]
\[ KB \]

Proof

\[ (X)d \land (X)e \land (a) \]
\[ (X)b \land (q) \land (u) \land (b) \]
\[ (X)\neg (X)d \]
\[ KB \]

Risk affected

\[ (X)d \land (X)e \land (a) \]
\[ (X)b \land (q) \land (u) \land (b) \]
\[ (X)\neg (X)d \]
\[ KB \]
Disjunctive Answers

- KB
- Delay literals of the form yes(X)
- Use top-down proof procedure
- We would like proof procedure to find yes(X)
  + Which you would have gotten if yes(a) and yes(b) was replaced with yes(a) ∧ yes(b)
  - Very different from answer that it has two solutions, one with X = a and another with X = b
  - Yes, with either X = a or X = b, but you don't know which

- Query p(X)

- Use top-down proof procedure
- Delay literals of the form yes(X)
- Stop when just yes atoms left in answer clause