
Overview

⇒ Integrity Constraints

- Disjunctive & Negative Knowledge
- Resolution Rule
- Bottom-Up
- Proof by Refutation
- Top-Down

Integrity Constraints (Chapter 7.3)

- *Integrity Constraint*

- *false* $\leftarrow a_1 \wedge \dots \wedge a_n$

- Means that $a_1 \wedge \dots \wedge a_n$ cannot be true

+ A model of KB must make each clause true

If it makes the body true, than 'false' must be true
Which is a contradiction, so body can't be true

- Allows us to specify things that should not be true

- *false* $\leftarrow a$ means that a has to be false in all models

- *Horn Clause*

- Is either a clause or an integrity constraint

- Have either an atom or a false on left hand side

Example

- KB
 $false \leftarrow a \wedge b$
 $a \leftarrow c$
 $b \leftarrow c$
 - 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 $a \wedge b$ true and hence would need to be make $false$ true
 - So $KB \models \neg c$

Syntax and Semantics of Not

- Add \neg to syntax
- Semantics
 - If interpretation I and variable assignment U 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* $\leftarrow a_1 \wedge \dots \wedge a_n$
 - Can be written as $\neg a_1 \leftarrow a_2 \wedge \dots \wedge a_n$
 - Can be written as $\neg a_1 \vee \dots \vee \neg a_n$
- *h* $\leftarrow a_1 \wedge \dots \wedge a_n$
 - Can be written as $h \vee \neg a_1 \vee \dots \vee \neg a_n$; **conjunctive normal form**
 - 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 \neg to syntax
- Semantics
 - If interpretation I and variable assignment U make a or b true, than it makes $a \vee b$ 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

Home Clauses

- Integrity constraints can be used for diagnostics
 - See textbook
- Also allows us a way to state some negative information
 - From home control: don't want something to be 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
- Top-Down

Disjunctive Knowledge

	In Conjunctive Normal Form	
	Positive literals	Negative literals
Datalog	Exactly 1	any number
Horne	At most 1	any number
Disjunctive & Negative	any number	any number

- Remove restriction of Horne clauses
 - Allow disjunction in head
 - $c \vee b \leftarrow a$
 - 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 \wedge , \vee , \neg , and \leftarrow

Conversion to Conjunctive Normal Form

- Any expression with \wedge , \vee , \neg , and \leftrightarrow can be converted into a set of clauses in conjunctive normal form
 - no literals on right hand side of \leftrightarrow
- **Step 1: Eliminate \leftrightarrow**
 - $(\phi \leftrightarrow \psi)$ replaced with $(\phi \vee \neg \psi)$
- **Step 2: Distribute negation so only applies to atoms**
 - $\neg \neg \phi$ replaced with ϕ
 - $\neg(\phi \vee \psi)$ replaced with $\neg \phi \wedge \neg \psi$
 - $\neg(\phi \wedge \psi)$ replaced with $\neg \phi \vee \neg \psi$
- **Step 3: Distribute \vee 's over \wedge 's**
 - $(\phi \vee (\psi \wedge \chi))$ replaced by $((\phi \vee \psi) \wedge (\phi \vee \chi))$

Example

$$\begin{aligned}
 &a \wedge (b \vee c \vee \neg(d \leftrightarrow e)) \\
 &a \wedge (b \vee c \vee \neg(d \vee \neg e)) \\
 &a \wedge (b \vee c \vee (\neg d \wedge \neg \neg e)) \\
 &a \wedge (b \vee c \vee (\neg d \wedge e)) \\
 &a \wedge (b \vee (c \vee (\neg d \wedge e))) \\
 &a \wedge (b \vee ((c \vee \neg d) \wedge (c \vee e))) \\
 &a \wedge ((b \vee (c \vee \neg d)) \wedge (b \vee (c \vee e))) \\
 &a \wedge (b \vee c \vee \neg d) \wedge (b \vee c \vee e)
 \end{aligned}$$

Conjunctive Normal Form

- Convert all clause into conjunctive normal form
 - Can view the literals as a set
 - + Duplicates are removed
 - + $a \leftarrow b \wedge c \wedge c$
 - + $a \vee \neg b \vee \neg c \vee \neg c$
 - + $\{a, \neg b, \neg c\}$
- Our KB is now a set of clauses,
where each clause is a set of literals

Overview

- Integrity Constraints
- Disjunctive & Negative Knowledge
 - \Rightarrow Resolution Rule
- Bottom-Up
- Proof by Refutation
- Top-Down

Resolution Rule

- **Resolution Rule**
 - Resolvent A includes $\neg a$
 - Resolvent B includes b
 - a and b can be unified
 - σ is the MGU of a and b
 - Let A' be A with $\neg a$ removed, and σ applied
 - Let B' be B with b removed, and σ 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
- Top-Down

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 $\neg A$, skip R, since trivially true
 - + If there is an $R' \in C$ such that $R' \subset R$, skip R, since already implied
 - + If there are any $R' \in C$ such that $R \subset R'$, remove R since now implied by R
- **Consequent set no longer just has atoms in it,**
but can have any arbitrary clause in disjunctive 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 σ such that $e\sigma$ is a subset of the query, then the query is true
- Is it sound and complete?

Example I

false $\leftarrow a \wedge b$

$a \leftarrow c$

$b \leftarrow c$

? $\neg c$

Example II

$$(a \vee \neg b) \leftarrow c$$
$$\neg e \leftarrow \neg c$$
$$b \vee d$$
$$(a \vee b) \leftarrow d$$
$$e \leftarrow \neg a$$
 $a \vee c$

Overview

- Integrity Constraints
 - Disjunctive & Negative Knowledge
 - Resolution Rule
 - Bottom-Up
- ⇒ Proof by Refutation
- Top-Down

Home Clauses and Resolution (not in textbook)

- What is so special about Home 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
 - Means KB has no model, which means it is inconsistent
- So if we want to prove q (where q is a literal)
 - Same as proving $KB \models q$
 - If this is the case, 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

Overview

- Integrity Constraints
- Disjunctive & Negative Knowledge
- Resolution Rule
- Bottom-Up
- Proof by Refutation
 - ⇒ Top-Down

Top Down Proof Procedure

- Start with query, which is a conjunction of literals
 - *yes* $\leftarrow p_1 \wedge \dots \wedge p_i \wedge \neg p_{i+1} \wedge \dots \wedge \neg p_n$
 - For previous top-down procedure, it was conjunction of positive literals
- Turn into disjunctive normal form
 - $\{ \textit{yes}, \neg p_1, \dots, \neg p_i, p_{i+1}, \dots, p_n \}$
- Use resolution rule to derive new answer clauses
 - Attack first non-yes literal in answer clause
 - Stop when just “yes” in answer clause

A problem

- Consider KB
 - $a \vee b$
 - $c \leftarrow a$ into disjunctive normal form \Rightarrow $a \vee b$
 - $c \leftarrow b$ $c \vee \neg a$
 - $c \vee \neg b$
- Proof
 - $?c$
 - $\textit{yes} \vee \neg c$ Use $c \vee \neg a$
 - $\textit{yes} \vee \neg a$ Use $a \vee b$
 - $\textit{yes} \vee b$ Use $a \vee b$
 - $\textit{yes} \vee c$ Use $c \vee \neg b$
 - 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

?c

yes∨¬*c* Use *c*∨¬*a* from KB

yes∨¬*a* Use *a*∨*b* from KB

yes∨*b*

Use *c*∨¬*b* from KB

yes∨*c*

Use *yes*∨¬*c* from prior answer clause in proof

yes

Disjunctive Answers

- KB

$p(X) \leftarrow q(X)$

$q(a) \vee q(b)$

$p(X) \vee \neg q(X)$

$q(a) \vee q(b)$

- 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 $q(a) \wedge q(b)$

- We would like proof procedure to find $yes(a) \vee yes(b)$

Solution

- Top-down proof procedure
 - Add answer clause to KB
 - + We might need to use it several times
 - Start with answer clause as usual
 - + (Slightly different from textbook)
- Use a delaying reasoning procedure, that delays literals of form $yes(X)$

Example

- KB

$$\begin{array}{l}
 p(X) \leftarrow q(X) \\
 q(a) \vee q(b) \vee q(c) \\
 yes(X) \leftarrow p(X)
 \end{array}
 \quad \text{into disjunctive normal form} \Rightarrow
 \quad \begin{array}{l}
 p(X) \vee \neg q(X) \\
 q(a) \vee q(b) \vee q(c) \\
 yes(X) \vee \neg p(X)
 \end{array}$$

- Proof

Answer Clause	How
$yes(X) \vee \neg p(X)$	$yes(X) \vee \neg p(X)$
$yes(X) \vee \neg q(X)$	$p(X) \vee \neg q(X)$
$yes(a) \vee q(b) \vee q(c)$	$q(a) \vee q(b) \vee q(c)$
$yes(a) \vee p(b) \vee q(c)$	$p(X) \vee \neg q(X)$
$yes(a) \vee yes(b) \vee q(c)$	$yes(X) \vee \neg p(X)$
$yes(a) \vee yes(b) \vee p(c)$	$p(X) \vee \neg q(X)$
$yes(a) \vee yes(b) \vee yes(c)$	$yes(X) \vee \neg p(X)$