Lecture 10 First-Order Logic Saturation-Based Reasoning

COMP24412: Symbolic AI

Giles Reger

February 2019

Aim and Learning Outcomes

The aim of this lecture is to:

Give the idea behind the completeness of resolution, introduce ordered resolution, and discuss clause selection

Learning Outcomes

By the end of this lecture you will be able to:

1. State what it means for an inference system to be (refutationally) complete
2. Describe the general idea behind the model construction approach
3. Describe, with examples, a fair clause selection approach
4. Apply the given clause algorithm with resolution (etc) to a set of clauses

First-Order Logic Stuff

Syntax (propositional logic with predicates and quantifiers)

Semantics in terms of models

Clausal representation

Reasoning with Clauses using Resolution

Reasoning with Equality with Paramodulation (and Equality Resolution)

Transformation to Clausal Form

Today

Ground resolution is sound and complete

The completeness argument allows us to optimise its application

We can lift it to first-order resolution

Need fairness, get given clause algorithm

Ground Resolution: Soundness

We consider the ground case. Reminder:

$$\frac{l \vee C \qquad \neg l \vee D}{C \vee D}$$

where l is a ground atom and C, D are ground clauses.

This rule is sound, we only derive true things.

For any model $\mathcal{M}$ if $\mathcal{M} \models l \vee C$ and $\mathcal{M} \models \neg l \vee D$ then $\mathcal{M} \models C \vee D$.

Two cases

1. $\mathcal{M} \models l$ and therefore $\mathcal{M} \models D$
2. $\mathcal{M} \models \neg l$ and therefore $\mathcal{M} \models C$

Note that $l \vee \neg l$ is a tautology.

Ground Resolution: Completeness

We consider the ground case. Reminder:

$$\frac{l \vee C \qquad \neg l \vee D}{C \vee D}$$

where l is a ground atom and C, D are ground clauses.

This rule is refutationally complete, if it is unsat we can show it.

Let N be a set of ground clauses and N^* be the set saturated with respect to the above rule. Then $N \models \mathit{false}$ if and only if $\mathit{false} \in N^*$.

If direction by soundness of resolution.

Only if direction by constructing a model of N from N^* if $\mathit{false} \notin N^*$.

Ordering Clauses

A partial ordering (irreflexive, transitive) $\succ$ is well-founded if there exist no infinite chains $a_0 \succ a_1 \succ a_2 \succ \ldots$

Assume a well-founded partial order $\succ$ on ground atoms. We could use a simple 'dictionary' order, which would also be total.

First, lift to literals such that $\neg l \succ l$ for every atom l

Now, lift to clauses: $C \succ D$ if for every l in D/C there is a $l' \succ l$ in C/D

Example, given $p \succ q \succ r$

$$p \vee q \succ p \succ \neg q \vee r \succ q \vee r$$

Observations on Clause Ordering

$\succ$ on clauses is total and well-founded

Let $\max(C)$ be the maximal literal in C, this exists and is unique

If $\max(C) \succ \max(D)$ then $C \succ D$

If $\max(C) = \max(D)$ but $\max(C)$ is neg and $\max(D)$ pos then $C \succ D$

This gives a stratification of clause sets by maximal literal

Stratified Clause Sets

Let $A \succ B$. Clause sets are then stratified in this form:

	B	$\dots \vee B$ $\dots$ $\dots \vee B \vee B$ $\dots$ $\neg B \vee \dots$	all D where $\max(D) = B$
$\prec$	$\vdots$	$\vdots$	
	A	$\dots \vee A$ $\dots$ $\dots \vee A \vee A$ $\dots$ $\neg A \vee \dots$ $\dots$	all C where $\max(C) = A$

Model Construction

Idea:

- Build up an interpretation $\mathcal{M}$ incrementally
- Look at clauses from smallest to largest
- If C is already true in I then carry on
- Otherwise, make the maximal literal in C true in I

We use $\mathcal{M}_C$ for the interpretation after processing C and Δ_C for the new clauses produced by C. Then

$$\begin{array}{rcl} \mathcal{M}_C & = & \bigcup_{C \succ D} \Delta_D \\ \Delta_C & = & \begin{cases} \{l\} & \textit{if } \mathcal{M}_C \not\models C \textit{ and } l = \max(C) \textit{ and } l \textit{ is pos} \\ \emptyset & \textit{otherwise} \end{cases} \end{array}$$

If $\Delta_C = \{l\}$ we say that C is produces l and C is productive

Example

Let $p_5 \succ p_4 \succ p_3 \succ p_2 \succ p_1 \succ p_0$

clauses	$\mathcal{M}_C$	Δ_C	Remark
$\neg p_0$	$\emptyset$	$\emptyset$	true in $\mathcal{M}_C$
$p_0 \vee p_1$	$\emptyset$	$\{p_1\}$	
$p_1 \vee p_2$	$\{p_1\}$	$\emptyset$	true in $\mathcal{M}_C$
$\neg p_1 \vee p_2$	$\{p_1\}$	$\{p_2\}$	
$\neg p_1 \vee p_3 \vee p_0$	$\{p_1, p_2\}$	$\{p_3\}$	
$\neg p_1 \vee p_4 \vee p_3 \vee p_0$	$\{p_1, p_2, p_3\}$	$\emptyset$	true in $\mathcal{M}_C$
$\neg p_1 \vee \neg p_4 \vee p_3$	$\{p_1, p_2, p_3\}$	$\emptyset$	true in $\mathcal{M}_C$
$\neg p_4 \vee p_5$	$\{p_1, p_2, p_3\}$	$\{p_5\}$	

So $\mathcal{M} = \{p_1, p_2, p_3, p_5\}$

Observations

We use these next:

If $C = \neg l \vee C'$ then C does not produce l and no $D \succ C'$ produces l

Therefore, if l in $\mathcal{M}$ then some smaller clause most produce l

If C is productive then $\Delta_C \models C$, hence $\mathcal{M} \models C$

Model Existence

Let $\mathcal{M}$ be the model constructed from N^* where *false* $\notin N^*$.
We have $\mathcal{M} \models N^*$.

Proof by contradiction. Suppose $\mathcal{M} \not\models N^*$.

There must be a smallest C in N^* s.t. $\mathcal{M} \not\models C$

C is not productive, hence $l = \max(C)$ is negative

$C = \neg l \vee C'$, hence $\mathcal{M} \not\models C'$ and $\mathcal{M} \models l$

As $\mathcal{M} \models l$ there is some $D = l \vee D'$, $C \succ D$ s.t. D produces l

So $\mathcal{M} \not\models D'$ (as D produces l)

By resolution, $C' \vee D' \in N^*$ and $\mathcal{M} \not\models C' \vee D'$

but $C \succ C' \vee D'$ thus C is not the smallest such clause

Compactness

Compactness of propositional logic follows.

A set of propositional formulas N is unsatisfiable if and only if there is a finite subset of N that is unsatisfiable.

The if part is non-trivial.

If N is unsatisfiable then N^* is unsatisfiable, thus $\mathit{false} \in N^*$

There must be a finite number of resolution steps required to derive *false*

Let P be the clauses in the resolution proof and $M = P \cap N$

M is finite, M is unsatisfiable and $M \subseteq N$

Ordered Resolution

Given the previous model construction we can observe that certain inferences can be excluded and the model construction still works.

If we resolve on either a maximal or negative literal then we do all inferences required by model construction.

This gives us ordered resolution:

$$\frac{l_1 \vee C \qquad \neg l_1 \vee D}{(C \vee D)\theta}$$

where l_1 is maximal in $l_1 \vee C$.

We're also allowed to arbitrarily select at least one negative literal in a clause and restrict inferences to the selected literals.

Less Work

Two choices of inference.

$$\neg \mathit{rich}(\mathit{giles}) \vee \mathit{happy}(\mathit{giles}) \qquad \mathit{rich}(\mathit{giles}) \qquad \neg \mathit{happy}(\mathit{giles})$$

Less Work

One choice.

$$\neg rich(giles) \vee happy(giles) \qquad rich(giles) \qquad \neg happy(giles)$$

Lifting

Now let N be a set of non-ground clauses.

Let $G_\Sigma(N)$ be the grounding of N using Σ

If N is saturated wrt non-ground resolution then $G_\Sigma(N)$ is saturated wrt ground resolution

We can apply the model construction with $G_\Sigma(N)$ and the result is a model of N

Compactness lifts in a similar way

We can do something similar with equality but much more work (and requires replacing paramodulation with something else)

Missing Rule

Usually we also have the (positive) factoring rule

$$\frac{C \vee l_1 \vee l_2}{\theta(C \vee l_1)} \; \theta = \text{mgu}(l_1, l_2)$$

Missing Rule

Usually we also have the (positive) factoring rule

$$\frac{C \vee l_1 \vee l_2}{\theta(C \vee l_1)} \ \theta = \mathrm{mgu}(l_1, l_2)$$

which is required in some cases

$$\begin{array}{lr} 1 & p(u) \vee p(f(u)) \\ 2 & \neg p(v) \vee p(f(w)) \\ 3 & \neg p(x) \vee \neg p(f(x)) \end{array}$$

Missing Rule

Usually we also have the (positive) factoring rule

$$\frac{C \vee l_1 \vee l_2}{\theta(C \vee l_1)} \ \theta = \text{mgu}(l_1, l_2)$$

which is required in some cases

$$\begin{array}{ll} 1 & p(u) \vee p(f(u)) \\ 2 & \neg p(v) \vee p(f(w)) \\ 3 & \neg p(x) \vee \neg p(f(x)) \end{array}$$

Resolvents

Missing Rule

Usually we also have the (positive) factoring rule

$$\frac{C \vee l_1 \vee l_2}{\theta(C \vee l_1)} \ \theta = \text{mgu}(l_1, l_2)$$

which is required in some cases

$$\begin{array}{ll} 1 & p(u) \vee p(f(u)) \\ 2 & \neg p(v) \vee p(f(w)) \\ 3 & \neg p(x) \vee \neg p(f(x)) \end{array}$$

Resolvents

$$\begin{array}{lll} 4 & p(u) \vee p(f(w)) & (1,2) \end{array}$$

Missing Rule

Usually we also have the (positive) factoring rule

$$\frac{C \vee l_1 \vee l_2}{\theta(C \vee l_1)} \; \theta = \mathrm{mgu}(l_1, l_2)$$

which is required in some cases

$$\begin{array}{lr} 1 & p(u) \vee p(f(u)) \\ 2 & \neg p(v) \vee p(f(w)) \\ 3 & \neg p(x) \vee \neg p(f(x)) \end{array}$$

Resolvents

$$\begin{array}{lrl} 4 & p(u) \vee p(f(w)) & (1,2) \\ 5 & p(u) \vee \neg p(f(f(u))) & (1,3) \end{array}$$

Missing Rule

Usually we also have the (positive) factoring rule

$$\frac{C \vee l_1 \vee l_2}{\theta(C \vee l_1)} \ \theta = \text{mgu}(l_1, l_2)$$

which is required in some cases

$$\begin{array}{lr} 1 & p(u) \vee p(f(u)) \\ 2 & \neg p(v) \vee p(f(w)) \\ 3 & \neg p(x) \vee \neg p(f(x)) \end{array}$$

Resolvents

$$\begin{array}{lrl} 4 & p(u) \vee p(f(w)) & (1,2) \\ 5 & p(u) \vee \neg p(f(f(u))) & (1,3) \end{array}$$

We're only going to get clauses with 2 literals.
However, we can factor (4) to $p(f(w))$
Resolving with (3) gives $\neg p(f(f(z)))$ then with $p(f(w))$ gives *false*

Fairness: Clause Selection

The above view is static, it assumes we have the saturated set

In reality we need to generate it but what if we infinitely delay performing an inference?

We lose the partial decidability

A saturation process is fair if no clause is delayed infinitely often

Two fair clause selection strategies:

- First-in first-out
- Smallest (in number of symbols) first
  (there are a finite number of terms with at most k symbols)

Given Clause Algorithm

input: *Init*: set of clauses;
var *active*, *passive*, *unprocessed*: set of clauses;
var *given*, *new*: clause;
active := $\emptyset$; *unprocessed* := *Init*;
loop
  while *unprocessed* $\neq \emptyset$
    new :=*pop*(*unprocessed*);
    if *new* $= \square$ **then** **return** *unsatisfiable*;
    add *new* to *passive*
  if *passive* $= \emptyset$ **then** **return** *satisfiable* or *unknown*
  given := *select*(*passive*); *(* clause selection *)*
  move *given* from *passive* to *active*;
  unprocessed :=*infer*(*given*, *active*); *(* generating inferences *)*

Complete Example

$$\left\{ \begin{array}{c} \forall x.(happy(x) \leftrightarrow \exists y.(loves(x,y))) \\ \forall x.(rich(x) \leftarrow loves(X, money)) \\ rich(giles) \end{array} \right\} \models happy(giles)$$