Comp24412 Symbolic AI

Lecture 8: Resolution theorem-proving

Ian Pratt-Hartmann

Room KB2.38: email: ipratt@cs.man.ac.uk

2016–17
• In an earlier lecture, we met SATCHMO, a first-order theorem-prover implemented in a single page of Prolog.
• It has nice termination properties, but it not practically very useful.
• This lecture introduces a more serious approach to constructing proofs in first-order logic: resolution.
• Resolution is the basis of many widely-used theorem-provers today.
• We would like to be able to determine logical relationships such as **validity**

• For example,

\[
\forall x (\text{artist}(x) \rightarrow \exists y (\text{beekeeper}(y) \land \text{admire}(x, y)))
\]

\[
\forall x (\text{beekeeper}(y) \rightarrow \forall y (\text{artist}(y) \land \text{admire}(y, x) \rightarrow \text{despise}(x, y)))
\]

\[
\forall x (\text{artist}(x) \rightarrow \exists y (\text{beekeeper}(y) \land \text{despises}(y, x) \land \text{admire}(x, y)))
\]
• Here is a more complex example

\[
\forall x \forall y \forall z (p(x, y) \land p(y, z) \rightarrow p(x, z))
\]

\[
\forall x \forall y \forall z (q(x, y) \land q(y, z) \rightarrow q(x, z))
\]

\[
\forall x \forall y (p(x, y) \rightarrow p(y, x))
\]

\[
\forall x \forall y (p(x, y) \lor q(x, y))
\]

\[
\forall x \forall y p(x, y) \lor \forall x \forall y q(x, y)
\]
• As with Satchmo, we first convert to clause form.
• A literal is an atomic formula or an atomic formula prefixed by \( \neg \).
• Thus, \( p, q(a), \neg r(x, y) \) are all literals; \( p \lor p, q(x) \land p \) and \( \neg \neg r(x, y) \) are not literals.
• A clause is a literal or a collection of literals all joined by \( \lor \).
• Thus \( \neg p, p \lor q(x), \neg p(a) \lor \neg r(a, b) \lor r \) are all clauses; \( p \land q(a), p \rightarrow p \) and \( \neg \neg r(x, y) \) are not clauses.
• The following result is easily established:

• Let $\varphi$ be a quantifier-free formula. Then there exist clauses $C_1, \ldots, C_n$ such that

\[ \models \varphi \leftrightarrow (C_1 \land C_2 \land \ldots \land C_n). \]

• In other words, any quantifier-free formula is logically equivalent to the conjunction of some collection of clauses.
• A formula in the predicate calculus in which all the quantifiers are at the front is said to be in **prenex form**.

• The formulas

\[ \forall x (\text{man}(x) \rightarrow \text{mortal}(x)) \]
\[ \forall x \exists y \text{loves}(x, y) \]

are in **prenex form**

• The formulas

\[ \forall x (\text{boy}(x) \rightarrow \exists y (\text{girl}(y) \land \text{loves}(x, y))) \]
\[ \forall x ((\text{boy}(x) \land \exists y (\text{girl}(y) \land \text{loves}(x, y))) \rightarrow \text{happy}(x)) \]

are not.
• The following result can easily be established:
• Let \( \varphi \) be a formula. Then there exists a prenex form formula \( \psi \) such that

\[
| = \varphi \leftrightarrow \psi.
\]

• In other words, any formula is logically equivalent to a formula with all the quantifiers at the front.
• Example

$$\forall x (\text{boy}(x) \rightarrow \exists y (\text{girl}(y) \land \text{loves}(x, y)))$$

is equivalent to

$$\forall x \exists y (\text{boy}(x) \rightarrow (\text{girl}(y) \land \text{loves}(x, y)))$$

• Example

$$\forall x ((\text{boy}(x) \land \exists y (\text{girl}(y) \land \text{loves}(x, y))) \rightarrow \text{happy}(x))$$

is equivalent to

$$\forall x \forall y ((\text{boy}(x) \land \text{girl}(y) \land \text{loves}(x, y)) \rightarrow \text{happy}(x))$$
• Given a prenex formula, existential quantifiers can be eliminated.

• Consider

\((\exists x)(\text{man}(x) \land \text{philosopher}(x))\)

• This is **equisatisfiable** with

\(\text{man}(a) \land \text{philosopher}(a).\)

(This name must be **new**: i.e. not occurring in any other formulas.)

• Similarly

\((\exists y)(\forall x)(\text{loves}(x, y))\)

is equisatisfiable with

\((\forall x)(\text{loves}(x, b)).\)
Things are different however, for:

\((\forall x)(\exists y)(\text{loves}(x, y))\).

This formula is equisatisfiable with:

\((\forall x)(\text{loves}(x, f(x)))\)

where \(f\) is a function (called a Skolem function).

Note the use of a function (rather than a constant): the formula says that everyone loves someone or other – but not necessarily the same person.

We need a function whenever we want to eliminate an existential quantifier which is to the right of some universal quantifiers.
• These new constants and functions are called **Skolem constants** and **Skolem functions**, respectively.

• The process of replacing existentially quantified variables with Skolem constants and functions is called **Skolemization**.

• Notice that Skolemization does not produce logically equivalent formulas; but it does produce equisatisfiable formulas.
• Suppose then, we put a formula $\varphi$ in prenex form, and Skolemize.

• The result will be

$$\forall x_1 \ldots \forall x_n \chi$$

where $\chi$ is quantifier-free.

• But the universal quantifiers convey no information any more, so we might as well write:

$$\chi$$

• Now we replace $\chi$ (the quantifier-free rump) by an equivalent collection of clauses.

• The result is said to be the clause form for $\varphi$. 
• So we can massage any formula of the predicate calculus into a collection of clause form expressions (containing variables) of the form

$$L_1 \lor \ldots \lor L_N$$

where the $L_i$ are literals

• Example:

$$\forall x (\text{boy}(x) \rightarrow \exists y (\text{girl}(y) \land \text{love}(x, y)))$$

has, as its clausal form,

$$\neg \text{boy}(x) \lor \text{girl}(f(x))$$

$$\neg \text{boy}(x) \lor \text{love}(x, f(x))$$

• Since we can equisatisfiably convert to clausal forms, we need only inference procedures which work on clausal forms.
• First, we look at the **ground** case.
• The rule **modus ponens**:

\[
p \rightarrow q \quad \frac{p}{q}
\]

• Re-writing:

\[
\neg p \lor q \quad \frac{p}{q}
\]

• Generalizing to longer clauses:

\[
\neg p \lor C \quad \frac{p \lor D}{C \lor D}
\]
• This gives us a way to make inferences.
• Consider the formulas
  \[ \text{boy}(\text{john}) \rightarrow (\text{girl}(\text{mary}) \rightarrow \text{loves}(\text{john}, \text{mary})), \]
  \[ \text{boy}(\text{john}), \quad \text{girl}(\text{mary}), \quad \neg \text{loves}(\text{john}, \text{mary}). \]
• It should be obvious that these formulas are not (simultaneously satisfiable)
• In clausal form:
  \[ \neg \text{boy}(\text{john}) \lor \neg \text{girl}(\text{mary}) \lor \text{loves}(\text{john}, \text{mary}), \]
  \[ \text{boy}(\text{john}), \quad \text{girl}(\text{mary}), \quad \neg \text{loves}(\text{john}, \text{mary}). \]
• We can apply resolution repeatedly as follows:

\[
\begin{align*}
girl(m) & \quad \neg boy(j) \lor \neg girl(m) \lor loves(j, m) \\
\text{boy}(j) & \quad \neg boy(j) \lor loves(j, m) \\
\neg loves(j, m) & \quad loves(j, m) \\
\hline
\bot.
\end{align*}
\]

• Since the empty clause \( \bot \) has been derived, it is evident (even to a computer) that the formulas we started with are not satisfiable.
• This gives us a way to test validity.

• For suppose we want to test whether

$$\varphi_1, \ldots, \varphi_n \models \psi.$$ 

• We convert

$$\varphi_1, \ldots, \varphi_n, \neg \psi$$

to clausal form and try to use resolution and factoring to obtain a contradiction. If we succeed, the sequent is valid.

• In particular, we have shown the validity of the argument with premises

$$\text{boy}(\text{john}) \rightarrow (\text{girl}(\text{mary}) \rightarrow \text{loves}(\text{john}, \text{mary})) \quad \text{boy}(\text{john}) \quad \text{girl}(\text{mary})$$

and conclusion $$\text{loves}(\text{john}, \text{mary}).$$
• Next, we look at the non-ground case.
• To apply resolution to non-ground clauses, we need the concept of unification.
• If $A$ and $A'$ are atoms, we say that $A$ and $A'$ are unifiable if there is a substitution $\sigma$ of terms for variables such that $A\sigma = A'\sigma$. 
• Example 1:

\[ \text{man}(x) \quad \text{man}(\text{socrates}) \]

unify, under the substitution \( \sigma : x \mapsto \text{socrates} \).

• Example 2:

\[ q(x, f(x)) \quad q(u, v) \]

unify, under the substitution \( \sigma : u \mapsto x; v \mapsto f(x) \).

• Example 3:

\[ p(x) \quad q(u) \]

do not unify.

• Example 4:

\[ p(x, f(x)) \quad p(f(y), y) \]

do not unify either.
• We noted that
  \[ q(x, f(x)) \quad q(u, v) \]
unify, under the substitution

  \[ \sigma : u \mapsto x; v \mapsto f(x) \]

making the common term

  \[ q(x, f(x)). \]

• But they also unify under the substitution, say,

  \[ \sigma : u \mapsto a; v \mapsto f(a); x \mapsto a, \]

making the common term

  \[ q(a, f(a)). \]
• If $A$ and $A'$ are unifiable, then there is a substitution $\theta$ such that $A\theta = A'\theta$ and, for any substitution $\sigma$ such that $A\sigma = A'\sigma$, we have $\sigma = \theta \rho$ for some substitution $\rho$.

• We say that $\theta$ is a **most general unifier** (m.g.u).

• The m.g.u. is unique up to renaming of variables.

• It can be efficiently computed . . .
• Now back to the analogue of modus ponens for non-ground clauses:

\[
\frac{\neg p(x) \lor q(x)}{p(a)}
\]

\[
\frac{q(a)}{}
\]

• Generalizing again:

\[
\frac{\neg A \lor C \quad A' \lor D}{(C \lor D)\theta}
\]

where \( A \) and \( A' \) are unifiable atoms with m.g.u. \( \theta \)

• This rule of inference is known as the resolution rule
• It is obvious that we can remove repeated literals from clauses:

\[
\frac{p \lor p \lor p \lor C}{p \lor C},
\]

• Generalizing to the non-ground case gives the **factoring** rule:

\[
\frac{C}{(C\theta)^0},
\]

with \(\theta\) the m.g.u of some literals in \(C\) and \(^0\) denoting deletion of repeated literals.
• Again, we can use resolution and factoring to construct proofs:

\[
\begin{align*}
girl(mary) & \quad \neg boy(x) \lor \neg girl(y) \lor lv(x, y) \\
boy(john) & \quad \neg boy(x) \lor lv(x, mary) \\
\neg lv(john, mary) & \quad lv(john, mary) \\
\hline
\bot.
\end{align*}
\]

• If clause \( C \) is derivable from a set \( C \) of clauses by means of resolution and factoring, we write \( C \vdash C \).
• As before, this gives us a way to make inferences.

• Consider the formulas

\[ \forall x (\text{boy}(x) \rightarrow \forall y (\text{girl}(y) \rightarrow \text{loves}(x, y))) \]  
\[ \text{boy}(\text{john}) \]  
\[ \text{girl}(\text{mary}) \]  
\[ \neg \text{loves}(\text{john}, \text{mary}) \].

• In clausal form:

\[ \neg \text{boy}(x) \lor \neg \text{girl}(y) \lor \text{loves}(x, y) \]  
\[ \text{boy}(\text{john}) \]  
\[ \text{girl}(\text{mary}) \]  
\[ \neg \text{loves}(\text{john}, \text{mary}) \].

• But these clauses imply \( \bot \).

• Therefore, the original set of formulas is unsatisfiable.
• As before: to test whether

\[ \varphi_1, \ldots, \varphi_n \models \psi, \]

• convert

\[ \varphi_1, \ldots, \varphi_n, \neg \psi \]

to clausal form and try to use resolution and factoring to obtain a contradiction.

• If you succeed, the sequent is valid
In particular, in showing that the set of formulas

$$\forall x(\text{boy}(x) \rightarrow \forall y(\text{girl}(y) \rightarrow \text{loves}(x, y))) \quad \text{boy}(\text{john})
\text{girl}(\text{mary}) \quad \neg\text{loves}(\text{john}, \text{mary}).$$

is unsatisfiable, we have shown that the argument

$$\forall x(\text{boy}(x) \rightarrow \forall y(\text{girl}(y) \rightarrow \text{loves}(x, y)))
\text{boy}(\text{john})
\text{girl}(\text{mary})
\text{loves}(\text{john}, \text{mary})$$

is valid.
• In fact, the following result can be proved:

**Theorem**

*Let $C$ be a set of clauses. Then $C \vdash \bot$ if and only if the universal closure of $C$ is unsatisfiable.*

• This means that resolution and factoring are all the rules we need to determine entailments

• We say that the proof system in question is (sound and) **complete**.

• To think about: Why does this not give us an algorithm for testing validity?
• Let $\mathcal{A}$ be the collection of all atoms (over some signature).
• Let $\prec$ be a partial order on $\mathcal{A}$.
• We say that $\prec$ is an $A$-ordering if, for all substitutions $\theta$, $A \prec A'$ implies $A\theta \prec A'\theta$.
• We extend any $A$-ordering $\prec$ to literals by ignoring negations.
• Example of A-ordering:

\[ A \prec^2 A' \text{ iff } A \text{ involves a unary predicate and } A' \text{ involves a binary predicate.} \]

• Example of A-ordering:

\[ A \prec^d A' \text{ iff } d(A) < d(A') \text{ and } d(x, A) < d(x, A') \text{ for all variables } x \text{ in } A, \]

where \( d(A) \) is the \textit{depth} of \( A \) and \( d(x, A) \) is the \textit{term depth} of \( x \) in \( A \).
• Given an A-ordering $\prec$, ordered resolution is the same as resolution, namely

$$
\begin{array}{c}
\neg A \lor C \\
A' \lor D
\end{array} \frac{}{(C \lor D)\theta},
$$

subject to the added restriction that, for every literal $L$ in $C$, $A \not\prec L$, and for every literal $L'$ in $D$, $A' \not\prec L'$.

• Similarly with ordered factoring.

• If clause $C$ is derivable from a set $C$ of clauses by means of $\prec$-ordered resolution and factoring, we write $C \vdash_\prec C$. 
• Amazingly, ordering restrictions never compromise completeness

**Theorem**

*Let $C$ be a set of clauses and $\prec$ an A-ordering. Then $C \vdash_{\prec} \bot$ if and only if $C$ is unsatisfiable.*
• Let us say that the **1-variable fragment** is the set of clauses $C$ satisfying either of the following conditions:
  
  • every literal of $C$ is ground;
  
  • there is a variable $x$ such that, for every literal $L$ of $C$, $\text{Vars}(L) = \{x\}$.

• The clauses

$$p(a) \lor q(b, f(a)), \quad p(x) \lor \neg r(h(x, g(x)), x)$$

are in the 1-variable fragment, but

$$p(a) \lor q(x, f(a)), \quad p(x) \lor \neg r(y, x)$$

are not in the 1-variable fragment.

• Note that resolution and factoring preserve membership in the 1-variable fragment.
• In general, resolution increases the functional depth of clauses:

\[
\frac{p(x) \lor q(fx) \quad \neg r(gx') \lor s(x') \lor \neg q(x')}{p(x) \lor \neg r(gfx) \lor s(fx)}
\]

• However, \(\prec_d\)-ordered resolution and factoring do not:

\[
\frac{p(x) \lor q(fx) \quad \neg r(x') \lor s(x') \lor \neg q(x')}{p(x) \lor \neg r(fx) \lor s(fx)}
\]
• Suppose we take a set $C$ of clauses in the 1-variable fragment, and keep applying resolution and factoring to them.

• Obviously, only clauses featuring the signature of $C$ will be generated.

• Because the functional depth is not increased, the total number of clauses that can be generated from $C$ by $≺_d$-ordered resolution and factoring is bounded by an exponential function of the size of the signature.

• Hence, the satisfiability of clauses in the 1-variable fragment (with a fixed depth-bound) can be algorithmically decided in exponential time.