Logic and theory of computation

5th lecture

15-10-2018

Prenex conjunctive normal form (PCNF)

PCNF

A formula is in **prenex conjunctive normal form (PCNF)** iff it is of the form: $Q_1 x_1 \cdots Q_n x_n M$ where the Q_i are quantifiers and M is a quantifier-free formula in CNF. The sequence $Q_1 x_1 \cdots Q_n x_n$ is the **prefix** and M is the **matrix**.

Example:
$\forall y \forall z((p(f(y)) \vee \neg p(g(z)) \vee q(z)) \wedge (\neg q(z) \vee \neg p(g(z)) \vee q(y)))$.

Clausal form of a formula in PCNF

Let A be a closed formula in PCNF whose prefix consists only of universal quantifiers. The clausal form of A consists of the matrix of A written as a set of clauses.

Example:
$\{\{p(f(y)), \neg p(g(z)), q(z)\}, \{\neg q(z), \neg p(g(z)), q(y)\}\}$.

Transforming a formula to PCNF

Theorem

For every closed formula A there is a formula A' in PCNF, such that $A \equiv A'$.

Steps of transformation (by a running example):
$\forall x(p(x) \to q(x)) \to (\forall x p(x) \to \forall x q(x))$.

• Rename bound variables so that no variable occurs in 2 quantifiers.
$\forall x(p(x) \to q(x)) \to (\forall y p(y) \to \forall z q(z))$.

• Eliminate implication.
$\neg\forall x(\neg p(x) \vee q(x)) \vee \neg\forall y p(y) \vee \forall z q(z)$.

• Push negation operators inward, collapsing double negation, until they apply to atomic formulas only. Use the equivalences:
$\neg\forall x A(x) \equiv \exists x \neg A(x), \quad \neg\exists x A(x) \equiv \forall x \neg A(x)$.

So we get: $\exists x(p(x) \wedge \neg q(x)) \vee \exists y \neg p(y) \vee \forall z q(z)$.

Transforming a formula to PCNF (cont'd)

• Extract quantifiers from the matrix. Choose an outermost quantifier, that is, a quantifier in the matrix that is not within the scope of another quantifier still in the matrix. Extract the quantifier using the following equivalences, where Q is a quantifier and op is either $\vee$ or $\wedge$:
$A \, op \, QxB(x) \equiv Qx(A \, op \, B(x)), \quad QxA(x) \, op \, B \equiv Qx(A(x) \, op \, B).$

Repeat until all quantifiers appear in the prefix and the matrix is quantifier-free. The equivalences are applicable since all the quantifiers name different variables. In the example, no quantifiers appear within the scope of another, so we can extract them in any order. E.g., in the order x, y, z:
$\exists x \exists y \forall z((p(x) \wedge \neg q(x)) \vee \neg p(y) \vee q(z)).$

• Use the laws of propositional logic to transform the matrix into CNF. The formula is now in PCNF.
$\exists x \exists y \forall z((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z))).$

Skolem's theorem

The notation $A \approx A'$ means that A' is satisfiable if and only if A is satisfiable; that is, there exists a model for A' if and only if there exists a model for A. This is not the same as logically equivalence $A \equiv A'$, which means that for all models $\mathcal{I}$, $\mathcal{I}$ is a model for A' if and only if it is a model for A.

Theorem(Skolem)

Let A be a closed formula. Then there exists a formula A' in clausal form such that $A \approx A'$.

We may assume that A is in PCNF.

Idea: In $A = \forall x \exists y p(x, y)$, the quantifiers can be read: for all x, **produce** a value y associated with that x such that the predicate p is true. Our intuitive concept of a function is the same: $y = f(x)$ means that given x, f produces a value y associated with x. The existential quantifier can be removed giving $A = \forall x p(x, f(x))$ with a **new** function symbol f.

Skolem's theorem (cont'd)

So, for every existential quantifier $\exists x$ in A, let $y_1, \ldots, y_n$ be the universally quantified variables preceding $\exists x$ and let f be a **new** n-ary function symbol. Delete $\exists x$ and replace every occurrence of x by $f(y_1, \ldots, y_n)$.

If there are no universal quantifiers preceding $\exists x$, replace x by a new constant (0-ary function). These new function symbols are called **Skolem functions** and the process of replacing existential quantifiers by functions is called **Skolemization**.

Skolem's theorem - Example 1

Example (cont'd):
$A = \exists x \exists y \forall z((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z)))$.

After Skolemization:

$A \approx \forall z((p(a) \vee \neg p(b) \vee q(z)) \wedge (\neg q(a) \vee \neg p(b) \vee q(z)))$,

where a and b are the Skolem functions (in our case: 0-ary functions, i.e. constants) corresponding to the existentially quantified variables x and y, respectively.

Which is in clausal form:

$\{\{p(a), \neg p(b), q(z)\}, \{\neg q(a), \neg p(b), q(z)\}\}$.

Skolem's theorem - Example 2

Example 2:

Consider again $\exists x(p(x) \wedge \neg q(x)) \vee \exists y \neg p(y) \vee \forall z q(z)$.

Let us extract the quantifiers in the order z, x, y in this case, the equivalent PCNF formula is:

$B = \forall z \exists x \exists y((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z)))$.

Since the existential quantifiers are preceded by a (single) universal quantifier, the Skolem functions are (unary) functions, not constants:

$B \approx \forall z((p(f(z)) \vee \neg p(g(z)) \vee q(z)) \wedge (\neg q(f(z)) \vee \neg p(g(z)) \vee q(z)))$,

Which is in clausal form:

$\{\{p(f(z)), \neg p(g(z)), q(z)\}, \{\neg q(f(z)), \neg p(g(z)), q(z)\}\}$

Skolem's theorem – Example 3

Example 3:

$$C = \exists x \forall y p(x, y) \rightarrow \forall y \exists x p(x, y) \equiv$$
$$\exists x \forall y p(x, y) \rightarrow \forall w \exists z p(z, w) \equiv$$
$$\neg \exists x \forall y p(x, y) \vee \forall w \exists z p(z, w) \equiv$$
$$\forall x \exists y \neg p(x, y) \vee \forall w \exists z p(z, w) \equiv$$
$$\forall x \exists y \forall w \exists z (\neg p(x, y) \vee p(z, w))$$

Skolemization: Remove existential quantifiers and introduce Skolem functions f and g:

$$C \approx \forall x \forall w (\neg p(x, f(x)) \vee p(g(x, w), w))$$

Which is in clausal form:

$$\{\{\neg p(x, f(x)), p(g(x, w), w)\}\}.$$

Herbrand universe

Herbrand universe

Let S be a set of clauses, and let $\mathcal{A}$ be the set of constant symbols and $\mathcal{F}$ be the set of function symbols appearing in S. $\mathcal{H}_S$, the **Herbrand universe** of S, is the following:

- $a_i \in \mathcal{H}_S$ for $a_i \in \mathcal{A}$,
- $f_i^0 \in \mathcal{H}_S$ for $f_i^0 \in \mathcal{F}$,
- $f_i^n(t_1, \ldots, t_n) \in \mathcal{H}_S$ for $n > 1, f_i^n \in \mathcal{F}$, $t_j \in \mathcal{H}_S$.

If there are no constant symbols or 0-ary function symbols in S, initialize $\mathcal{H}_S$ with an arbitrary constant symbol a.

Examples:

$S_1 = \{\{p(a), \neg p(b), q(z)\}, \{\neg p(b), \neg q(z)\}\}$, $\mathcal{H}_{S_1} = \{a, b\}$

$S_2 = \{\{\neg p(x, f(y))\}, \{p(w, g(w))\}\}$

$\mathcal{H}_{S_2} = \{a, f(a), g(a), f(f(a)), g(f(a)), f(g(a)), g(g(a)), \ldots\}$

$S_3 = \{\{\neg p(a, f(x, y))\}, \{p(b, f(x, y))\}\}$

$\mathcal{H}_{S_3} = \{a, b, f(a, a), f(a, b), f(b, a), f(b, b), f(a, f(a, a)), \ldots\}$

Ground ...

A **ground term** is a term which does not contain any variables.
A **ground atomic formula** is an atomic formula, all of whose terms are ground.
A **ground literal** is a ground atomic formula or its negation.
A **ground formula** is a quantifier-free formula, all of whose atomic formula are ground.
A is a **ground instance** of a quantifier-free formula A' iff it can be obtained from A' by substituting ground terms for the (free) variables in A'.

Example:
The terms $a, f(a,b), g(b, f(a,b))$ are ground.
$p(f(a,b), a)$ is a ground atomic formula and $\neg p(f(a,b), a)$ is a ground literal.
$A = p(f(x,y), a)$ is not a ground atomic formula because of the variables x, y.
$p(f(a, f(a,b)), a)$ is a ground instance of A.

Herbrand base

Herbrand base

Let H_S be the Herbrand universe for S. $\mathcal{B}_S$, the **Herbrand base** for S, is the set of ground atomic formulas that can be formed from predicate symbols in S and terms in $\mathcal{H}_S$.

Example:
$S_3 = \{\{\neg p(a, f(x, y))\}, \{p(b, f(x, y))\}\}$

We've seen that
$\mathcal{H}_{S_3} = \{a, b, f(a, a), f(a, b), f(b, a), f(b, b), f(a, f(a, a)), \ldots\}$

p is the only predicate symbol in S_3.

$$\begin{aligned}\mathcal{B}_{S_3} = \{ \quad & p(a, f(a, a)),\ p(a, f(a, b)),\ p(a, f(b, a)),\ p(a, f(b, b)), \ldots, \\ & p(a, f(a, f(a, a))), \ldots, \\ & p(b, f(a, a)),\ p(b, f(a, b)),\ p(b, f(b, a)),\ p(b, f(b, b)), \ldots, \\ & p(b, f(a, f(a, a))), \ldots\}.\end{aligned}$$

Herbrand interpretation

Informally, a *Herbrand interpretation* for a formula in clausal form S is the following:

- the domain is $\mathcal{H}_S$,
- the constants of S are mapped identically,
- the function symbols of S are mapped identically and interpretted naturally (for example f is mapped to f and say, $f(a, f(a, a))$ is defined to be $f(a, f(a, a)))$,
- Herbrand interpretations differ only in the interpretation of predicates, which is arbritrary. The only restriction is that arities should match.

Observe, that each Herbrand interpretation can be associated with a subset J of the Herbrand base. Let p be an arbitrary predicate symbol in S and R be the relation corresponding to p. An interpretation can be associated with J as follows. If $p(t_1, \ldots t_n) \in J$, then let $R(t_1, \ldots t_n) = T$. If $p(t_1, \ldots t_n) \notin J$, then let $R(t_1, \ldots t_n) = F$.

Herbrand interpretation – Example

Example:

$S_3 = \{\{\neg p(a, f(x, y))\}, \{p(b, f(x, y))\}\}$

$\mathcal{H}_{S_3} = \{a, b, f(a, a), f(a, b), f(b, a), f(b, b), f(a, f(a, a)), \ldots\}$

$$\begin{aligned}\mathcal{B}_{S_3}\{= \quad & p(a, f(a, a)), p(a, f(a, b)), p(a, f(b, a)), p(a, f(b, b)), \ldots, \\ & p(a, f(a, f(a, a))), \ldots, \\ & p(b, f(a, a)), p(b, f(a, b)), p(b, f(b, a)), p(b, f(b, b)), \ldots, \\ & p(b, f(a, f(a, a))), \ldots\}.\end{aligned}$$

Let
$J = \{p(a, f(a, b)), p(b, f(a, b)), p(b, f(b, a)), p(b, f(a, f(a, a)))\}.$

J defines an Herbrand interpretation.

$$\begin{aligned}v(p(a, f(a, b))) = \quad & v(p(b, f(a, b))) = v(p(b, f(b, a))) = \\ & v(p(b, f(a, f(a, a))))) = T\end{aligned}.$$

$v(r) = F$ for the infinite other $r \in \mathcal{B}_{S_3}$.

Herbrand model

If $\mathcal{I} \models S$ holds for a Herbrand interpretation $\mathcal{I}$ then $\mathcal{I}$ is called an **Herbrand model** for S.

Theorem

A set of clauses S has a model iff it has an Herbrand model.

Herbrand's Theorem

A set of clauses S is unsatisfiable if and only if a finite set of ground instances of clauses of S is unsatisfiable.

Example: The clausal form of the formula:
$\neg[\forall x(p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x))]$ is:

$S = \{\{\neg p(x), q(x)\}, \{p(y)\}, \{\neg q(a)\}\}$ (a is a Skolem constant).

A set of ground instances obtained by substituting a for each variable is: $S' = \{\{\neg p(a), q(a)\}, \{p(a)\}, \{\neg q(a)\}\}$.

Clearly, S' is unsatisfiable. Herbrand's Theorem states that the unsatisfiability of S' implies that S is unsatisfiable.

Semi-decision procedure for validity

From Herbrand's theorem we obtain a relatively efficient semi-decision procedure for validity of formulas in first-order logic:

1. Negate the formula;
2. Transform into clausal form;
3. Generate a finite set of ground clauses;
4. Check if the set of ground clauses is unsatisfiable.

We have already discussed the first two steps. For the last one any convenient decision procedure for the propositional logic can be used by treating each distinct ground atomic formula as a distinct propositional letter. Unfortunately, we have no efficient way of generating a set of ground clauses that is likely to be unsatisfiable.

Ground resolution

Resolvent

Let C_1, C_2 be ground clauses such that $\ell \in C_1$ and $\ell^c \in C_2$. C_1, C_2 are said to be *clashing clauses* and to clash on the complementary literals ℓ, ℓ^c. C, the *resolvent* of C_1 and C_2, is the clause:

$$\text{Res}(C_1, C_2) = (C_1 - \{\ell\}) \cup (C_2 - \{\ell^c\}).$$

C_1 and C_2 are the parent clauses of C.

Lemma

The resolvent C is satisfiable if and only if the parent clauses C_1 and C_2 are both satisfiable.

Ground resolution procedure can be defined similarly to resolution procedure in propositonal logic.

Theorem

Ground resolution procedure is a sound and complete refutation procedure for first order logic.

Ground resolution (cont'd)

Example:

$S' = \{\{\neg p(a), q(a)\}, \{p(a)\}, \{\neg q(a)\}\}$.

S' is unsatisfiable. Take the resolvent of the first two clauses resulting $\{q(a)\}$. Then take the resolvent of this and the third clause resulting $\{\square\}$.

Ground resolution is not a useful refutation procedure for first-order logic because the set of ground terms is infinite (assuming that there is even one function symbols).

Robinson (1965) showed that how to perform resolution on clauses that are not ground by looking for substitutions that create clashing clauses.

The definitions and algorithms are rather technical so we skip it, but let us see an example.

General resolution (example only)

1.	$\{\neg p(x), q(x), r(x, f(x))\}$	
2.	$\{\neg p(x), q(x), r'(f(x))\}$	
3.	$\{p'(a)\}$	
4.	$\{p(a)\}$	
5.	$\{\neg r(a, y), p'(y)\}$	
6.	$\{\neg p'(x), \neg q(x)\}$	
7.	$\{\neg p'(x), \neg r'(x)\}$	
8.	$\{\neg q(a)\}$	$3, 6[x \leftarrow a]$
9.	$\{q(a), r'(f(a))\}$	$2[x \leftarrow a], 4$
10.	$\{r'(f(a))\}$	$8, 9$
11.	$\{q(a), r(a, f(a))\}$	$1[x \leftarrow a], 4$
12.	$\{r(a, f(a))\}$	$8, 11$
13.	$\{p'(f(a))\}$	$5[y \leftarrow f(a)], 12$
14.	$\{\neg r'(f(a))\}$	$7[x \leftarrow f(a)], 13$
15.	$\{\square\}$	$10, 14$

Summary

Ground resolution and general resolution procedures are sound and complete refutation procedures for validity in first order logic.

General resolution procedure is much more effective than ground resolution and has proved to be a successful method for automated theorem proving in first-order logic. ($\rightarrow$ logic programming, Prolog)

Not even general resolution is a decision procedure for validity in first order logic.

Can we do something better?

NO. Church proved that validity is undecidable for first order logic.
($\rightarrow$ 2nd part, theory of computation)