Transformation of First-Order Formulas to Sets of Clauses

Hans de Nivelle, Marc Bezem

First-Order Logic

Terms are defined as follows:

- If f is a function symbol with arity n, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Atoms are defined as follows:

- If p is a predicate symbol with arity n, and $t_1, \ldots, t_n$ are terms, $p(t_1, \ldots, t_n)$ is an atom.

Note: We have no formal distinction between variables and constants. When a 0-arity function symbol is bound, it is a variable. Otherwise, it is a constant.

First-Order Logic

The set of formulas is recursively defined as follows:

- $\bot$ and $\top$ are formulas,
- every atom is a formula,
- if F is a formula, then $\neg F$ is a formula,
- If F and G are formulas, then
  $F \vee G, \quad F \wedge G, \quad F \rightarrow G, \quad F \leftrightarrow G$ are formulas.
- If F is a formula, and v is a variable, then $\forall v\ F$ and $\exists v\ F$ are formulas.

First-Order Logic

Definition: Let F be a first-order formula, v be a variable. We call v free in F if there is an occurrence of v in F which is not in the scope of a $\forall v$ or $\exists v$.

Definition: Two formulas F and G are called α-variants of each other, if they differ only in bound variables.

Definition: Let F be a first-order formula, v be a variable, t be a term. We define capture avoiding substitution $F[v := t]$ (somewhat informally) as follows:

- As long as there is a free occurrence of v in F, which is in the scope of a quantifier $\forall x$ or $\exists x$, s.t. x occurs in t, replace in F the bound variable x by another variable y that does not occur in t.
- After that, simply replace all free occurrences of v by t.

Theorem Proving Procedure:

Resolution is a refutation-based procedure.

That means that resolution can only establish unsatisfiability.

Suppose we want to test whether $A_1, \ldots, A_p \vdash B$.

Then we try to refute $A_1, \ldots, A_p, \neg B$.

Before we can use resolution, we have to replace $A_1, \ldots, A_p, \neg B$ by clauses.

This is done through a series of transformations that we now will describe:

Negation Normal Form

Definition A formula is in negation normal form if it has no occurrences of $\rightarrow$ and $\leftrightarrow$, and $\neg$ is applied only on atoms.

Algorithm for NNF

The first parameter is the polarity, either 0 or 1. the second parameter is the formula.

- For an atom A,
  $\mathrm{NNF}(0, A) = \neg A$,
  $\mathrm{NNF}(1, A) = A$,

- $\mathrm{NNF}(0, \top) = \bot$,
  $\mathrm{NNF}(1, \top) = \top$,

- $\mathrm{NNF}(0, \bot) = \top$,
  $\mathrm{NNF}(1, \bot) = \bot$,

Algorithm for NNF (2)

- $\mathrm{NNF}(0, P \vee Q) = \mathrm{NNF}(0, P) \wedge \mathrm{NNF}(0, Q)$,
  $\mathrm{NNF}(1, P \vee Q) = \mathrm{NNF}(1, P) \vee \mathrm{NNF}(1, Q)$,

- $\mathrm{NNF}(0, P \wedge Q) = \mathrm{NNF}(0, P) \vee \mathrm{NNF}(0, Q)$,
  $\mathrm{NNF}(1, P \wedge Q) = \mathrm{NNF}(1, P) \wedge \mathrm{NNF}(1, Q)$,

- $\mathrm{NNF}(0, P \to Q) = \mathrm{NNF}(1, P) \wedge \mathrm{NNF}(0, Q)$,
  $\mathrm{NNF}(1, P \to Q) = \mathrm{NNF}(0, P) \vee \mathrm{NNF}(1, Q)$,

Algorithm for NNF (3)

- $\mathrm{NNF}(0, P \leftrightarrow Q) =$
  $(\ \mathrm{NNF}(1, P) \wedge \mathrm{NNF}(0, Q)\ ) \vee (\ \mathrm{NNF}(0, P) \wedge \mathrm{NNF}(1, Q)\ ),$
  $\mathrm{NNF}(1, P \leftrightarrow Q) =$
  $(\ \mathrm{NNF}(0, P) \vee \mathrm{NNF}(1, Q)\ ) \wedge (\ \mathrm{NNF}(1, P) \vee \mathrm{NNF}(0, Q)\ ),$

- $\mathrm{NNF}(0, \exists x\ P) = \forall x\ \mathrm{NNF}(0, P),$
  $\mathrm{NNF}(1, \exists x\ P) = \exists x\ \mathrm{NNF}(1, P),$

- $\mathrm{NNF}(0, \forall x\ P) = \exists x\ \mathrm{NNF}(0, P),$
  $\mathrm{NNF}(1, \forall x\ P) = \forall x\ \mathrm{NNF}(1, P),$

Skolemization

Skolemization replaces existential quantifiers by function symbols.

If we have a formula F_1 (in NNF) containing an existential quantifier, then write F_1 in the form $F_1 = [\ \exists y\ P(\overline{x}, y)\ ]$.

1. $P(\overline{x}, y)$ is an existentially quantified subformula, that is not in the scope of another existential quantifier.
2. $\overline{x}$ are the variables that are free in P, (not counting y) and that are bound by a universal quantifier outside $P(\overline{x}, y)$.

F_1 is replaced by $F_2 = F[\ P(\overline{x}, f(\overline{x}))\ ]$, where f is a function symbol that occurs nowhere else.

It is easy to see that $F_2 \models F_1$, but $F_1 \models F_2$ does not hold.

Skolemization

It can be shown that if F_1 has a model, then F_2 has a model. A (very sketchy) argument is as follows:

Let $(D, [\ ])$ be the interpretation that makes F_1 true. Here D is the domain.

For every $d \in D$, we define the value $f(d)$ as follows:

- If there is an $e \in D$, s.t. $(d, e) \in [P]$, then possibly there exists more than one e, s.t. $(d, e) \in [P]$ holds. Choose (in some mysterious way) one of those e, and make it the value of $f(d)$.
- If there is no $e \in D$, s.t. $(d, e) \in [P]$ holds, then give $f(d)$ an aribrary value. (Surprise!, here we are not using AC, because we can reuse the same element all the time)

(A real proof takes 5 or more pages. If you want to avoid using the of axiom of choice, even longer)

Skolemization, Another Definition

We first define $\mathrm{SKOL}(F)$ as $\mathrm{SKOL}(\{\}, F)$. Then:

- For a literal L, $\quad \mathrm{SKOL}(V, L) = L$,
- $\mathrm{SKOL}(V, \top) = \top$,
- $\mathrm{SKOL}(V, \bot) = \bot$,
- $\mathrm{SKOL}(V, P \vee Q) = \mathrm{SKOL}(V, P) \vee \mathrm{SKOL}(V, Q)$,
- $\mathrm{SKOL}(V, P \wedge Q) = \mathrm{SKOL}(V, P) \wedge \mathrm{SKOL}(V, Q)$,
- $\mathrm{SKOL}(V, \forall x\ P) = \forall x\ \mathrm{SKOL}(V \cup \{x\}, P)$,
- $\mathrm{SKOL}(V, \exists y\ P)$ is defined as $\mathrm{SKOL}(V, P[y := f(x_1, \ldots, x_n)])$.

  Here $x_1, \ldots, x_n$ is an enumeration of the variables that are free in P and which occur in V.

  f is a function symbol of arity n that occurs nowhere else.

Factoring of Disjunctions

We first introduce some notation:

Definition: Let S_1 and S_2 be sets of formulas. The product $S_1 \times S_2$ is defined as:

$$\{(s_1 \vee s_2) \mid s_1 \in S_1 \text{ and } s_2 \in S_2\}.$$

(The semantics of a set of formulas is defined as follows:

Definition:

$\bigwedge\{\ \} = \top$,
$\bigwedge\{F\} = F$,
$\bigwedge\{F_1, \ldots, F_n\} = F_1 \wedge \cdots \wedge F_n$, if $n > 2$.)

Factoring of Conjunctions

- For a literal L, $\quad \mathrm{FACT}(L) = \{L\}$,
- $\mathrm{FACT}(\top) = \emptyset$,
- $\mathrm{FACT}(\bot) = \{\bot\}$,
- $\mathrm{FACT}(P \wedge Q) = \mathrm{FACT}(P) \cup \mathrm{FACT}(Q)$,
- $\mathrm{FACT}(P \vee Q) = \mathrm{FACT}(P) \times \mathrm{FACT}(Q)$,
- $\mathrm{FACT}(\forall x\ P)$ is defined as:

$$\mathrm{FACT}(\forall x\ P) = \{(\forall x\ F) \mid F \in \mathrm{FACT}(P)\}.$$

Final Step:

At this point, we have a sequence of formulas, which is in NNF, contains no $\top$, no $\wedge$, and no $\exists$. Each formula F in the sequence is replaced by $\mathrm{CLS}(F)$, which is defined as follows:

- For a literal L, $\quad \mathrm{CLS}(L) = \{L\}$,
- $\mathrm{CLS}(\bot) = \{\ \}$,
- $\mathrm{CLS}(P \vee Q) = \mathrm{CLS}(P) \cup \mathrm{CLS}(Q)$,
- $\mathrm{CLS}(\forall x\ P) = \mathrm{CLS}(\ P[x := X]\ )$, where X is an implicit variable that occurs nowhere else.

Subformula Replacement

We now have described a clause transformation, which works well on most formulas, but which may cause an exponential increase other other formulas. There are two reasons for exponential increase:

- Many nested implications:

$$A_1 \leftrightarrow (A_2 \leftrightarrow (A_3 \leftrightarrow \cdots (A_{n-1} \leftrightarrow A_n) \cdots)).$$

- Disjunctions containing many subconjunctions:

$$(A_1 \wedge B_1) \vee (A_2 \wedge B_2) \vee \cdots \vee (A_n \wedge B_n).$$

Solution: Replace problematic subformulas in advance.

Subformula Replacement (2)

The subformula replacement function REPL uses a set Π which represents the positions of F that have to be replaced. In case it replaces a subformula P, it uses the variables V to determine the quantified variables that P depends on. The function REPL collects the set of definitions in a global variable Δ.

In case $P \in \Pi$, $\quad \mathrm{REPL}(\Pi, P, V, \Delta)$ is defined as follows: Let $x_1, \ldots, x_n$ be the variables that occur in V and that are free in P. Let p be a new predicate symbol of arity n. First compute $Q = \mathrm{REPL}'(\Pi, P, V, \Delta)$. Then add

$$\forall x_1, \ldots, x_n \;\; p(x_1, \ldots, x_n) \leftrightarrow Q$$

to Δ. Return $\mathrm{REPL}(\Pi, p, V, \Delta) = p(x_1, \ldots, x_n)$.

Subformula Replacement (3)

In case $P \notin \Pi$, then $\mathrm{REPL}(\Pi, P, V, \Delta) = \mathrm{REPL}'(\Pi, P, V\Delta)$, which is defined as follows:

- $\mathrm{REPL}'(\Pi, \bot, V, \Delta) = \bot, \quad \mathrm{REPL}'(\Pi, \top, \Delta) = \top,$
- For a literal L, $\mathrm{REPL}'(\Pi, \bot, V, \Delta) = L,$
- For a formula of form $\neg P$,
  $\mathrm{REPL}'(\Pi, \neg P, V, \Delta) = \neg\ \mathrm{REPL}'(V, P, V, \Delta)),$
- For a formula of form $P \times Q$, $\mathrm{REPL}'(\Pi, P \times Q, V, \Delta) =$
  $REPL'(\Pi, P, V, \Delta) \times \mathrm{REPL}'(\Pi, Q, V, \Delta)).$

  Here $\times$ is one of $\wedge,\ \vee,\ \rightarrow,\ \leftrightarrow$.
- For a formula of form $(\forall\exists)v\ P$,
  $\mathrm{REPL}'(\Pi, (\forall\exists)v\ P, V, \Delta) = REPL'(\Pi, (\forall\exists)v\ Q, V, \Delta).$

Summary

Now you understand how to efficiently translate a set of formulas into a set of clauses.