Harvey Mudd College  
Computer Science 80  
Logic for Computer Science  
Fall Semester 1999

Project #1 – Propositional Logic: Implementing Resolution  
Phase 1: Conversion to CNF  
Due 5:00pm, Thursday November 11, 1999

The purpose of this project is to implement the first phase of a resolution refutation theorem prover. In particular, you will implement the conversion to Conjunctive Normal Form. The project is to be implemented in either rex or SML. I have provided datatype/representation specifications below for both languages.

For this phase you are to implement two functions: cnf, and cnf_list. The first takes a representation of a formula and converts it to conjunctive normal form, represented as a list of lists of literals. The second does the same, but for a list of formulas, representing a conjunction of those formulas. (Its output is therefore the concatenation of the lists resulting from converting each one individually.)

The definition of cnf_list should be written in terms of cnf, and should be only a couple of lines long. The cnf function does almost all the work. You will find it helpful to implement cnf in terms of a group of support functions each of which implements one phase of the conversion.

Rex Specification

If you are using rex then formulas are to be represented using strings and lists as follows:

- ⊥ — "false"
- ⊤ — "true"
- p (a propositional letter) — "p"
- ¬A — ["NOT", A]
- A ∧ B — [A,"AND",B]
- A ∨ B — [A,"OR",B]
- A ⇒ B — [A,"IMPLIES",B]
- A ≡ B — [A,"EQUIV",B]
You will find it useful during testing to define the following variables.

\[
\begin{align*}
    & a = "a"; \\
    & b = "b"; \\
    & c = "c"; \\
    & d = "d"; \\
    & \texttt{not} = "\texttt{NOT}"; \\
    & \texttt{and} = "\texttt{AND}"; \\
    & \texttt{or} = "\texttt{OR}"; \\
    & \texttt{imp} = "\texttt{IMPLIES}"; \\
    & \texttt{equ} = "\texttt{EQUIV}";
\end{align*}
\]

and to similarly define variables for any other propositional letters you use in you tests. This way you can write \([a, \texttt{or}, [\texttt{not}, b]]\) instead of \(["a", "\texttt{or}", ["\texttt{not}", "b"]\).

**SML Specification**

If you are using SML for your solution, you should use the following \texttt{datatype} to store formulas:

\[
\begin{align*}
    \texttt{datatype} & \quad \texttt{wff} = \texttt{Bot} \\
    & \quad | \quad \texttt{Top} \\
    & \quad | \quad \texttt{Atom} \quad \texttt{of} \quad \texttt{string} \\
    & \quad | \quad \texttt{Not} \quad \texttt{of} \quad \texttt{wff} \\
    & \quad | \quad \texttt{And} \quad \texttt{of} \quad \texttt{wff} \times \texttt{wff} \\
    & \quad | \quad \texttt{Or} \quad \texttt{of} \quad \texttt{wff} \times \texttt{wff} \\
    & \quad | \quad \texttt{Implies} \quad \texttt{of} \quad \texttt{wff} \times \texttt{wff} \\
    & \quad | \quad \texttt{Equiv} \quad \texttt{of} \quad \texttt{wff} \times \texttt{wff};
\end{align*}
\]

The two required functions should have the type:

\[
\begin{align*}
    & \texttt{val} \quad \texttt{cnf} = \texttt{fn} : \texttt{wff} \rightarrow \texttt{wff} \texttt{list} \texttt{list} \\
    & \texttt{val} \quad \texttt{cnf\_list} = \texttt{fn} : \texttt{wff} \texttt{list} \rightarrow \texttt{wff} \texttt{list} \texttt{list}
\end{align*}
\]

During testing it will save you some typing if you make the following definitions:

\[
\begin{align*}
    & \texttt{val} \quad a = \texttt{Atom} \ "a"; \\
    & \texttt{val} \quad b = \texttt{Atom} \ "b"; \\
    & \texttt{val} \quad c = \texttt{Atom} \ "c"; \\
    & \texttt{val} \quad d = \texttt{Atom} \ "d";
\end{align*}
\]

and to similarly define variables for any other propositional letters you use in you tests. This way you can write \(\texttt{Or}(a, \texttt{Not}(b))\) instead of \(\texttt{Or}(\texttt{Atom} "a", \texttt{Not}(\texttt{Atom} "b"))\).

You’ll also find it helpful to issue the following two commands at the beginning of your source code, so that the system will echo back more of a given data structure.

\[
\begin{align*}
    & \texttt{Compiler\_Control\_Print.printDepth} := 100; \\
    & \texttt{Compiler\_Control\_Print.printLength} := 100;
\end{align*}
\]