Propositional Logic: Implementing Resolution
Phase 2: Resolution Refutation

Due: 5:00pm, Friday December 13, 2002
With No-Penalty Extension to: 12:00pm, Wednesday December 18
No further extensions or use of late-days allowed

The purpose of this phase of the project is to implement the core of the propositional resolution refutation theorem prover: the actual resolution engine. The project is to be implemented in SML. There are two principal tasks involved.

Task 1
You must write the function resolve, which, given two clauses (i.e. two lists of literals), returns a list of all the clauses that can result from any single application of the resolution rule. For example, if you give it the pair:

\[ [a, c, \neg d] \quad [b, \neg c, e] \]

It would produce the singleton list of clauses:

\[ [[a, b, \neg d, e]] \]

since they can only be resolved in one way. But if you gave it the pair:

\[ [a, \neg b, c, \neg d] \quad [b, \neg c, e] \]

It would produce the list of clauses:

\[ [[a, b, \neg d, e], [a, c, \neg c, \neg d, e]] \]

since there are two ways to resolve the pair of clauses. Finally, if you give it the pair:

\[ [a, c, \neg d] \quad [b, e] \]

It would produce the empty list of clauses:

\[ [] \]

because the two clauses cannot be resolved, as they contain no clashing literals. Thus the type of this function is wff list -> wff list -> wff list list.
Note that producing the empty list of clauses is not the same as producing an empty clause (which we call box). If you give the function the pair:

\[ [c] \quad [\neg c] \]

It would produce the singleton list of clauses, where the one clause is the empty clause:

\[ [\square] \]

**Task 2**

You must implement one main function: consequence, which, given a list of formulas and a single formula, proves whether the single formula is a logical consequence of the list of formulas. The type of this function is \texttt{wff list -> wff -> bool}.

To accomplish this, consequence should first convert each of the formulas in the list to CNF (by calling \texttt{cnf list} from the first phase), and convert the negation of the single formula to CNF (by calling \texttt{cnf} from the first phase). It should then put all the resultant clauses in a single list and send them to the function \texttt{refute} which attempts to build a resolution refutation of the set of clauses. Thus the type of \texttt{refute} is \texttt{wff list list -> bool}.

This project should be submitted using \texttt{cs80submit} as assignment 2. You should submit just the functions \texttt{resolve}, \texttt{consequence}, and \texttt{refute}, and any support functions you write for them. While these functions call \texttt{cnf list}, and \texttt{cnf} You should not include those functions. We will test your submission using the sample solutions for those functions. Your file also should not include (directly, or by use) the \texttt{wff} type definition.

The sml binary at \texttt{/cs/cs80/sml/sml-cs80} has been updated to include the functions \texttt{cnf} and \texttt{cnf list}, which you can call from your code.

**Extra Credit**

There are several extra-credit options:

- (15%) When refutation succeeds, return a data structure from which the refutation proof can be extracted.
- (15%) When refutation fails, return a satisfying valuation for the set of clauses.
- (5% each) Implement one of the (pre-)optimizations of resolution discussed in class.

You should note in the header of your submission for the project which, if any, extra-credit portions you are attempting.