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

Projects #2 & #3 – Propositional Logic: Implementing Resolution
Phase 2: Resolution Refutation
Sub-Phase 1 – Due 5:00pm, Tuesday November 23, 1999
Sub-Phase 2 – Due 5:00pm, Thursday December 2, 1999

The purpose of this project is to implement the core of the propositional resolution refutation theorem prover: the actual resolution engine. The project is to be implemented in `rex` or ML. In order to insure that you use your time for this project wisely, it has been further divided into two sub-phases. Each will count towards half the grade.

Sub-Phase 1

For the first sub-phase, 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:

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

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:

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

It would produce the list of clauses:

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

since there are two ways to resolve the pair of clauses.

This phase should be submitted using `cs80submit` as project 2. You should submit just the function `resolve` and any support functions. You should not include code for the conversion to CNF, which is not relevant to this phase. This sub-phase must be submitted by midnight Wednesday November 24. Submission between the due time and that time will count as a single late day.

Sub-Phase 2

For this phase you are to 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.

To accomplish this, `consequence` should first convert each of the formulas in the list to CNF (by calling `cnf_list` from the last phase), and convert the negation of the single formula to CNF ((by calling `cnf` from the last phase). It should then put all the resultant clauses in a single list and send them to the function `refute` which attempts to build a resolution refutation of the set of clauses. (For those working in `rex`, `refute` and `consequence` should return `0` if the refutation fails, and `1` if it succeeds in deriving box (the empty clause). For those working in ML, return the appropriate `bool` value.)

This phase should be submitted using `cs80submit` as project 3. You should submit just the functions `refute` and and `consequence` and any support functions. While these functions call `resolve`, `cnf_list`, and `cnf` You should not include those functions. We will test your submission using the sample solutions for those functions.

There are several extra-credit options for the second sub-phase of the project:

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

You should note in the header of your submission for project 3 which extra-credit portions you are attempting.