Propositional Theorem Prover

CSE-504, Spring 2012

Homework 1

Sandeep Kumar Koppala
Karthik Uthaman

This document presents the overall design of the code for Propositional Theorem Prover that we developed for Homework 1.

Overall Strategy:

There are two parts to the problem.

1. Syntax Checking / Validation
2. Consistency Checking / Satisfiability

Syntax Checking:

- The input statement is checked for the presence of a "." (dot / period) symbol in the end. If it is not present, an error is thrown.

- Notice that "->" is the only operator with two characters. So, the correct usage of these two symbols is validated (as they cannot be a part of any identifier otherwise). An error is reported in case of incorrect usage. Otherwise, all the appearances of "->" are replaced with "-". This 'preprocessing' helps in handling the statement better as all the terminals are single characters.

- The whole statement is divided into sub-parts based on the first terminal symbol that is encountered, and each part is checked recursively. E.g. If an '&' is encountered, the left part should be valid statement and the right part should also be a valid statement.

- This recursive approach takes care of the unary and binary operators.

- A statement, which contains no terminal symbols is assumed to be an identifier. It is checked to see if it contains only alphanumeric characters.

Consistency:

- When a statement passes the syntax check, it is scanned to find out all the different identifiers in it. For each identifier encountered, it is checked if it has been already encountered. To do so, a list of unique identifiers encountered so far is maintianed.

- A truth table is also maintained, which contains all possible combinations of truth values for all the identifiers.

- When a new identifier is encountered, it is added to the existing list of identifiers and also added to the truth table with all possible combinations updated.

- Now, the statement is converted from its inherent infix form to the corresponding postfix form.

- Once the postfix form is ready, iterate through all the rows in the truth table and replace the identifiers in the postfix form with the corresponding truth value in the row.

- This postfix form (with the truth values) is passed to a function which evaluates it using a stack and returns the value.

- For any row in the truth table, if the current statement evaluates to *False*, then the row is immediately removed from the table.

- After iterating through all the rows in this fashion, if there is at least one row in the truth table, then output “*Ok*” and accept the next statement. If the truth table becomes empty, display “*No*” and exit.