1 Overview

This lab is a continuation of Lab 06, where you worked recursively with propositions as BST-like structures. In this lab, you will build off the work done in Part 1 to realize the end goal of turning any proposition into CNF!

**Be sure to download and work in the provided code!** It contains correct implementations of the functions you wrote last week – which you will need for this week’s tasks.

2 Forms Refresher

First, spend a few minutes reviewing the writeup for Lab 06.

Next, let’s recap the different proposition forms, and the related functions in Build.java:

- **Simplified Form**: contains only ∨ and ¬ operators. The toSimplified function you implemented last week converts any Proposition to simplified form.
  
  Example simplified proposition: ¬(q ∧ s) ∧ ¬(¬r)

- **Negation Normal Form (NNF)**: contains only ∧, ∨, and ¬ operators. Additionally, any negation connectives (¬) can only be applied to atomic propositions (i.e. AtomicProp nodes). The simplifiedToNNF() function you implemented last week converts a simplified Proposition to NNF.
  
  Example NNF proposition: (q ∧ ¬s) ∨ ¬r

- **Conjunctive Normal Form (CNF)**: a proposition that is the conjunction of one or more clauses, where each clause is the disjunction of some number of (possibly negated) literals. Simply put, a CNF proposition is the and of a bunch of or clauses. This week’s NNFtoCNF() function converts an NNF proposition to CNF.
  
  Example CNF proposition: ¬q ∧ (r ∨ ¬s ∨ t) ∧ (q ∨ ¬t)

- **Disjunctive Normal Form (DNF)**: essentially the inverse of CNF – a proposition that is the disjunction of one or more clauses, where each clause is the conjunction of some number of (possibly negated) literals (the or of a bunch of and clauses). There is no function to convert to DNF, but you should still be aware of its definition.
  
  Example DNF proposition: (p ∧ t) ∨ q ∨ (¬s ∧ ¬t ∧ p)

3 Your Task

Your task is to finish the implementation of the final two functions, NNFtoCNF and toCNF, which will ultimately allow you to convert any Proposition to CNF. The details of your two tasks are on the next page:
3.1 From NNF to CNF

Your first task, which will be the bulk of the lab, is to finish the implementation of `NNFtoCNF`. This function accepts an argument `Proposition` in NNF and converts it to CNF. `NNFtoCNF` is fully implemented for you, you only need to write the recursive helper function `partialCNFtoCNF`.

The `partialCNFtoCNF` recursive helper that you will implement is similar to `NNFtoCNF` – its end goal is to ensure that the argument `phi` is in CNF. However, unlike `NNFtoCNF`, `phi` must at least partially already be in CNF, the specifics of which you will uncover in the pre-lab questions below.

Lastly, `partialCNFtoCNF` should not need to call `NNFtoCNF`, and you cannot modify the provided `NNFtoCNF`.

3.2 To CNF

Finally, the moment you have been waiting for! Implement `toCNF` which converts any `Proposition` to CNF.

4 Pre-Lab Questions

1. Review the mystery proposition trees below, and answer the two questions below:

   (a) One mystery tree is a CNF proposition and the other is a DNF proposition. Which is which? (hint: try writing out the propositions)

   (b) Based on the above, we know a proposition tree is CNF if no __________ is a descendant of a __________.

2. Review the below table relating the distributive law to proposition logic:

   **Principle of Distributivity**
   
   - $A \land (B \lor C) \equiv (A \land B) \lor (A \land C)$
   - $A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$
Convert the following propositions to CNF:

- \( p \lor (r \land q) \)
- \( p \lor (r \land q \land s) \)
- \((p \land t) \lor (r \land q)\)

3. Review the provided \texttt{NNFtoCNF} function. If neither of the first two conditionals are true, what type of proposition must \( \phi \) be?

4. Trace the rest of \texttt{NNFtoCNF}. The \texttt{partialCNFtoCNF} function requires its argument \( \phi \) to be \textit{at least} partially CNF. What part(s) of \( \phi \) can you assume are already CNF?

   \textit{hint: remember the black-box logic in your tracing of \texttt{NNFtoCNF}}

5. Given your response to questions 1b and 4, how do you know if \( \phi \) is not \textit{CNF}, and thus requires conversion?

6. Given your response to question 1b, is it possible for the root of a proposition tree to be a \textit{conjunct}, and for the tree to be \textit{CNF}?

5 Submission

See the top of this document for the lab’s due date and time. When submitting your code, include \textit{only} the files listed below:

- \texttt{Build.java}
- \texttt{Lab07Tester.java}