Jennifer Storozum
Elizabeth Wells
Computational Semantics
Final Project Report
Branching Futures & Computational Tree Logic

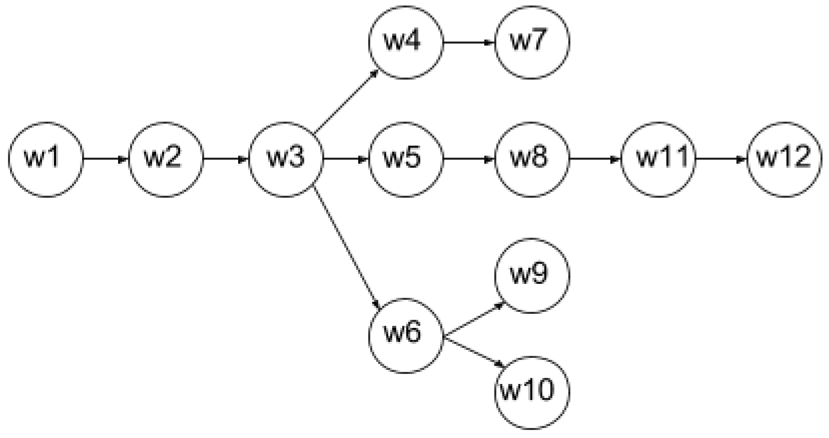
Your submission should also contain a write-up describing how you attacked this problem, what avenues you explored, and what problems you ran into. There is no length requirement, but you should include some use cases for testing and you should describe your strategy, process, and assumptions in some depth.

## Our Approach

We chose to expand on the work done for assignment 5 on LTL. Originally, we planned to split up our CTL operators into two groups, one for the operators over all paths (A and E) and another for the operators over a specific path (X, F, W, and G), and then compose the two with some kind of function to make well-formed formulas in CTL. We abandoned that idea early on when we couldn't come up with a way to compose two types of the same data type. In retrospect, we probably could have done it if we had made A and E different data types than X, F, W and G.

For the sake of convenience, we added a parameter *name* to the World data type. Also, because W takes two propositions as arguments, the parameter *propositions* in the World data type had to be modified to be a list of Props instead of a single Prop. We added a few extra worlds to our model, and populated the worlds with propositions in the present tense.

Our model looks like this:



Then we set about modifying the isSatisfied function to handle the CTL temporal operators, as the isSatisfiable and isValid functions rely on isSatisfied to most of the heavy computational lifting.

In the isSatisfied function, we list all the possible temporal operators and define the truth conditions for each one. An important assumption that we made is that the current world is not included in the path starting from that world. These are the interpretations of the temporal operators that we based our code on:

EX : phi holds in at least one of the next worlds from current world
EW: : there exists at least one path starting from the current world where phi holds until psi is true
EF : phi is eventually true somewhere along any path starting from, but not including, the current world
EG : there exists one path starting from, but not including the current world, where phi is true in all worlds along path
AX : phi holds in all the next worlds from the current world
AW: phi holds until psi is true in all paths starting from the current world

These conditions are complex to asses, so we wrote a variety of helper functions:

- nextWorlds, which returns a list of the worlds to the immediate right of the given world
- untilOr and untilAnd, recursive functions which are used to move the computation for EW and AW out of isSatisfied
- findAllPaths and findAllPathsAux, which return a list of all the paths starting from the given world (not inclusive)
- flatten, which returns and flattened list
- inWorld, which returns a boolean value indicating if a certain proposition is in a given world

Luckily during this process, we discovered the Haskell keywords *and* and *or* which shorthand for folds of lists of booleans and we adjusted our isValid and isSatisfiable code accordingly to make it clearer.

For the second part of the project, we decided to implement logical operators:

- NOT - logical negation
- AND - logical conjunction
- OR - logical disjunction (inclusive)
- ARROW - logical implicature/conditional
- BICONDITIONAL - logical biconditional

Test Set

Our full test set consists of 14 regular propositions and 6 pairs of propositions combined with Boolean operators and is available in our code right under the world definitions. There is at least one true example and at least one false for every pair of temporal operators and every boolean operation. Here are some examples:

- test1 = CTLProp AG ["the dwarves believe snowwhite"]

    - isSatified test1 w1 = True
    - isValid test1 = True
    - isSatisfiable test1 = True
- test3 = CTLProp EG ["some girls admire princesses"]
    - isSatified test3 w1 = True
    - isValid test3 = False
- test5 = CTLProp AX ["this dwarf lives"]
    - isSatified test5 w4 = True
    - isSatified test5 w5 = False
- test9 = CTLProp AF ["many princesses know atreyu"]
    - isSatisfied test9 w2 = True
    - isSatisfied test9 w11 = False
- test13 = CTLProp EW ["several girls see the giant", "those boys admire atreyu"]
    - isSatisfied test13 w1 = True
- test15 = CTLLogProp AG "the dwarves believe snowwhite" AND "some girls admire princesses"
    - isSatisfied test15 w5 = True
    - isSatisfied test15 w1 = False
- test18 = CTLLogProp EX "few giants sit" ARROW "several girls see the giant"
    - isSatisfied test18 w2 = True
    - isSatisfied test18 w6 = False