COSI135: Fall 2014
Assignment #5

This assignment builds on the vocabulary you added to the parser in the last assignment. Your task is to build a "tense interpreter" that can handle a large lexicon, such as the one you created, and multiple tense/aspect combinations.

Before you begin, you should take the provided file `World.hs` and add it to the folder containing the modified parser suit from your last assignment. We are only providing this one file, so that your tense interpreter will work with you lexicon and your parser, and with any edits you may have made to those. `World.hs` is the new entry point. In GHCI, run `:l World.hs`, and that will load `HRAS` and all dependent modules. You should be able to run any parser code as you have been, though in some cases, you may need to prepend the module name to the function call (e.g. `P.process`).

`World.hs` contains 5 worlds, representing different time indices. Given an arbitrary world as the "now," you can consider the world to the left to be the immediate past, the world to the left of that to be the remote past, the world to the right of the now to the immediate future, and the world to the right of that to be the distant future. Any worlds to the right of the distant future are simply times further in the future, and any worlds to the left of the remote past are simple times further in the past.

It is strongly recommended that you create a development test set of propositions with which to populate your worlds. A process had been in development to automate this for you, but it proved to be too time-consuming to have everyone implement it before starting the assignment. You should therefore manually generate 20-30 sentences that can be parsed by your parser, using a wide span of verbs, and distribute them randomly over the five worlds in `World.hs` by adding those strings to the "propositions" fields of the worlds. **These propositions should only use verbs in the present tense.**

There should be about 10 propositions per world, and there should definitely be some propositions used in multiple worlds. You should assume a closed world: if a proposition is not true (i.e. stated) in a world, it should be considered to be false in that world.

There are four temporal operators that can combine with a proposition: H, P, F, and G, defined as follows:

1. H — the global past. $H\phi$ means that ϕ is true at all past worlds.
2. P — somewhere in the past. $P\phi$ means that ϕ is true at at least one past world. $H\phi$ necessarily entails $P\phi$.
3. F — somewhere in the future. $F\phi$ means that ϕ is true at at least one future world. Under the modal interpretation used here, F is considered to also contain the present, so if ϕ is true now, $F\phi$ is also true.
4. G — the global future. $G\phi$ means that ϕ is true at all future worlds (including the present, as above). $G\phi$ necessarily entails $F\phi$.

`World.hs` contains a datatype `TProp`, which is the combination of a temporal operator (datatype `TemporalOperator`) and a proposition (datatype `String`).

You need to implement four functions: `isValid`, *isSatisfiable*, *isSatisfied* and *entailments*.:

- `isValid` — determine if a temporal proposition is valid in this model. That is, if given F *"John eats"*, `isValid` should return true if F *"John eats"* is true at all worlds. In this example, if *"John eats"* is true at *w5*, F *"John eats"* would be valid in the model since *w5* is the future-most world.
- `isSatisfiable` — determine if a temporal proposition is satisfiable in this model. That is, if given F(*"John eats."*), `isSatisfiable` should return true if F *"John eats"* is true at at least one world (if there is some world where "John eats" is true in the future or present).
- `isSatisfied` — determine if a temporal proposition is satisfied at the given world. That is, if given F *"John eats"* at *w3*, `isSatisfied` should return true if *"John eats"* is true at *w3*, *w4*, **or** *w5*. Meanwhile G *"John eats"* at *w3* would return true for *isSatisfied* only if *"John eats"* is true at *w3*, *w4*, **and** *w5*.

- `entailments` — determine what a particular proposition would entail if true at a particular world. *"John ate"* at *w3* would entail that *"John eats"* would have to be true at either *w1* or *w2*, but you don't know anymore than that. However, if *"John ate"* is true at *w2*, you know that *"John eats"* would have to be true at *w1*, since we assume there are no worlds outside the model. The modal operators `Necessarily` and `Possibly` (datatype `ModalOperator`) have been provided to help you handle uncertainty.

You should think about what effect using different verb tenses will have on the validity and satiability of the proposition. For instance, if *"John eats"* is true at *w4*, does that mean that F *"John will eat"* is true at *w2*. If *"John ate"* is true at *w1*, does that mean that P *"John has_eaten"* is true at *w3*? How about at *w2*? We will test your code with a variety of propositions using sentences that can be generated from your lexicon, so your code should be robust enough to handle any case that your parser can parse.

You should submit your modified parser suite (include all files, even ones you didn't make changes to). Please submit a compressed archive (.zip, .tar, .rar, or other compressed formats), and not individual files. You may work in pairs if you choose. If so, please only submit one assignment for your group.