IFT 6755 -H2019 | PROJECT 1: PROPOSITIONAL LOGIC

The project's due date is Friday March 1st, 24:00 AoE. Submit your solutions on StudiUM as a ZIP file containing any relevant PDFs, text, and source files. Your ZIP file should also contain a README.txt describing exactly where I can find the answer for each part. For Part 1, it is best if you typeset your answers with LaTeX. If you choose to do the exercises by hand, make sure your handwriting is very clear (no cursive!) and submit a legible scan of your solution. Yes, the 4 parts of the project add up to 105%.

PART 1 (35%)

Solve the following exercises from the book (Huth & Ryan, Logic in Computer Science, Second Edition):

1. Page 78, Exercises 1.1.1 a, d, h, i
2. Page 78, Exercises 1.2.1[1] c, f, h, j, l, r, x
3. Page 79, Exercises 1.2.2 b, c, e, g
4. Page 79, Exercises 1.2.3 n,q
5. Page 87, Exercise 1.4.17 a, b
6. Page 89, Exercise 1.5.15 a, g
7. Page 90, Exercise 1.5.17

(For proofs of validity, you can't use automated reasoners.)

PART 2 (10%)

For this Task, you can use the Sat4j solver (http://www.sat4j.org/howto.php#standalone). Get the core JAR from here: http://download.forge.ow2.org/sat4j/sat4j-core-v20130525.zip. To run the SAT solver with DIMACS input, follow these instructions: http://www.sat4j.org/howto.php#standalone

1. Construct the CNF formulas for all 3 parts of the Exercise 1.5.7 in page 88 of the book.
2. Construct the CNF formula for Exercise 1.5.9 in page 88 of the book.
3. Express the 4 formulas from the two previous questions in the DIMACS format. (See slide 38 of the Propositional Logic slides)
4. Check their satisfiability with Sat4j
5. Report the valuation found by Sat4j in terms of the propositional variables in steps 1 and 2.

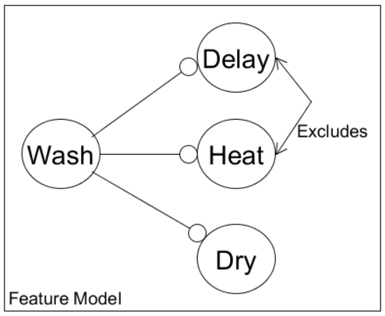
PART 3 (10%)

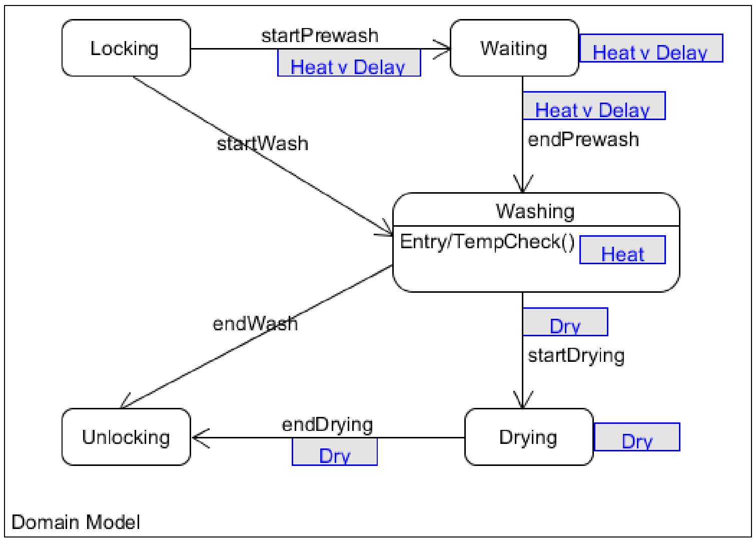
Use the Z3 solver to complete Exercise 1.2.2 in page 79 of the book. You can use the online interface (https://rise4fun.com/Z3/), but I recommend installing it (https://github.com/Z3Prover/z3/releases) and figuring out how to run it from the command line. See the Z3 tutorial (https://www.rise4fun.com/z3/tutorial) for the syntax for creating propositional logic queries. Explain how you transformed the sequents into Z3 queries and why the Z3 result is a proof of whether each sequent is valid.

[1] I am not talking about Exercise 1.1.2!

PART 4 (50%)

Software Product Lines (SPLs) are a methodology for modelling and managing families of related software products that are similar but have slight variations. Consider this example of an SPL **W** of washing machine controllers:





The SPL consists of three parts:

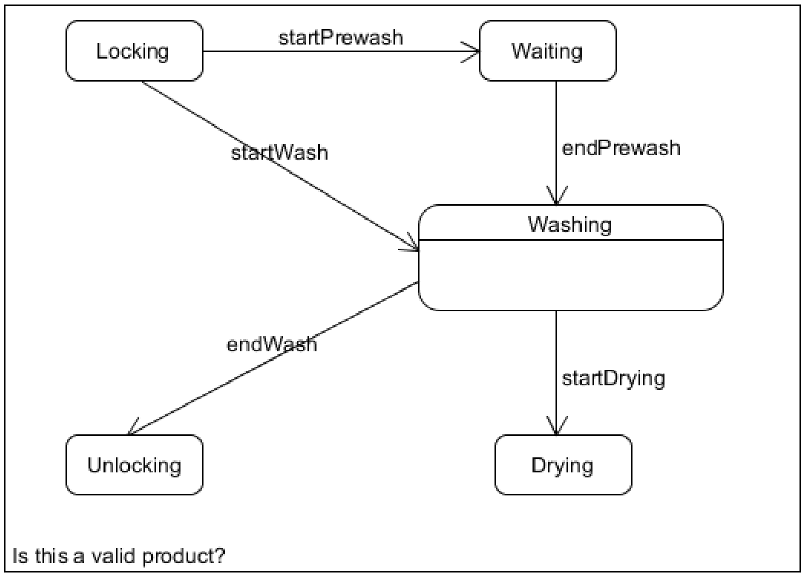
- The *feature model* **F** of **W** allows for three optional features to be added to a basic washing machine: **Heat** adds the ability to have hot water washes, **Dry** adds an automatic dry following the wash, and **Delay** adds the ability to delay the start time of the wash. Note that the heated wash and delayed wash features are mutually exclusive while drying can be added independently. The *Excludes* constraint between **Heat** and **Delay** in the feature model indicates that at most one of these can be selected. A set of features that form a valid combination is called a *configuration*.
- The *domain model* **D** of **W** is a statemachine which specifies that after initiating and locking the washer, a basic wash begins or a prewashing waiting period is initiated, either for heating the water or for a delayed wash. Then the washing takes place, followed, optionally, by drying. Finally,if drying or heating was used, the clothes are cooled and the washer is unlocked, terminating the process.
- Depending on which of the features have been selected for a particular washing machine product, only some parts of this state machine controller will be available. The propositional formulas in boxes throughout the controller state machine indicate the *presence conditions* **C** for different model elements, i.e., the configurations of features under which the element is present in a product. For example, the transition *startPrewash* from state *Locking* to state *Waiting* is only present if the feature **Heat** or the feature **Delay** is selected. Creating a particular washing machine product from **W** is called *configuring* the SPL.

Tasks:

1. Encode the feature model **F** of **W** in propositional logic using Z3.
   - Hint: you can omit the default feature **Wash**
2. Prove that your encoding forbids configurations that violate the mutual exclusion constraint between **Delay** and **Heat**.
3. Encode the domain model **D** of **W** in propositional logic using Z3. Treat *Entry/TempCheck()* as a single model element.

- Hint 1: each model element should be represented by a propositional variable meaning "the element is present".
- Hint 2: an edge can only be present in the model if both its endpoints are also present in the model.
- Hint 3: a contained element can only be present in the model if its container is also present.
- Hint 4: don't bother with encoding the metamodel and the typing of model elements.

4. Encode the set of presence conditions **C** of **W**.
    - Hint: a presence condition is merely a constraint between feature and model element variables
    - Hint: you can omit the default feature **Wash**
5. Combine **F**, **D**, **C**, and verify that your encoding of **W** is satisfiable.
6. Using Z3, check whether the following model can be generated by configuring **W** or not.


Locking
startPrewash
Waiting
startWash
endPrewash
Washing
endWash
startDrying
Unlocking
Drying
Is this a valid product?