EE570—Artificial Intelligence
Fall 2015
Problem Set #2
Due: Friday 25–September–2015

This problem set is an exercise in *interpretation*. It will teach you about *semantics* and *evaluation*. From an electrical-engineering perspective, you will build a simple logic simulator. From an AI perspective, you will define the semantics of propositional logic by way of an evaluator or interpreter for propositional logic.

Propositional logic is a language defined as follows. **true** and **false** are *truth values*. You have a set $\mathcal{P}$ of *propositions* (aka variables) p, q, r, ... A *formula* Φ is either

- a truth value,
- a proposition,
- $(\neg\Phi)$,
- $(\Phi_1 \wedge \Phi_2)$, or
- $(\Phi_1 \vee \Phi_2)$.

When writing formulas in mathematical notation, we adopt the convention that $\wedge$ has higher precedence than $\vee$ and optionally eliminate parentheses when it does not change the parse of a formula.

A *binding* $p \mapsto t$ maps the proposition p to the truth value t. A *truth assignment* I is a set of bindings. A truth assignment *maps* p to t if it contains $p \mapsto t$. A truth assignment is *consistent* if does not map any proposition to both **true** and **false**. A truth assignment is *complete* for Φ if it maps every proposition in Φ to a truth value. A truth assignment is *redundant* for Φ if it maps some proposition that is not in Φ to a truth value.

A *valuation function* $\mathcal{V}(\Phi, I)$ assigns a truth value to a formula Φ given a complete consistent truth assignment for Φ. Defining the valuation function specifies the semantics of propositional logic. We adopt the standard definition of $\mathcal{V}(\Phi, I)$ as follows:

$$\begin{aligned}
\mathcal{V}(t, I) &\triangleq t \\
\mathcal{V}(p, I) &\triangleq t \text{ when } p \mapsto t \in I \\
\mathcal{V}(\neg\Phi, I) &\triangleq \neg\mathcal{V}(\Phi, I) \\
\mathcal{V}(\Phi_1 \wedge \Phi_2, I) &\triangleq \mathcal{V}(\Phi_1, I) \wedge \mathcal{V}(\Phi_2, I) \\
\mathcal{V}(\Phi_1 \vee \Phi_2, I) &\triangleq \mathcal{V}(\Phi_1, I) \vee \mathcal{V}(\Phi_2, I)
\end{aligned}$$

A *row* for Φ is a pair $\langle I, t\rangle$ where I is a complete consistent nonredundant truth assignment for Φ and $t = \mathcal{V}(\Phi, I)$. The *truth table* for Φ is the set of all rows for Φ.

We will represent the truth values **true** and **false** as the SCHEME values `#t` and `#f` respectively. We will represent propositions as SCHEME symbols. We will represent the formulas $\neg\Phi$, $(\Phi_1 \wedge \cdots \wedge \Phi_n)$, and $(\Phi_1 \vee \cdots \vee \Phi_n)$, as the SCHEME S-expressions `(not` Φ`)`, `(and` Φ_1 `...` Φ_n`)`, and `(or` Φ_1 `...` Φ_n`)` respectively. We will represent the binding $p \mapsto t$ as the SCHEME list `(`p t`)`. We will represent sets as SCHEME lists. Thus we will represent a truth assignment like $\{p \mapsto \textbf{true}, q \mapsto \textbf{false}\}$ as `((p #t) (q #f))`. We will represent the row $\langle I, t\rangle$ as the SCHEME list `(`I t`)`.

We want you to implement the following procedure:

`truth-table` Φ [*Procedure*]

Φ is a formula. Returns the truth table for Φ.

To help debug and test your implementation, we have provided the GUI `(p2)`. The GUI allows you to create and edit formulas and interactively display their truth tables. The GUI has a *mode* which you can set by clicking on the buttons T, F, P, NOT, AND, and OR. It also has a parameter k which you can decrement and increment by clicking on the buttons −K and +K respectively. The GUI displays a formula. Initially it is *empty* When you click on a formula or subformula, that formula or subformula is replaced with a new formula of type mode. New T and F formulas generate formulas that are truth values. New P formulas generate the proposition p_k. New AND and OR formulas have arity k. Whenever the formula does not contain any empty subformulas, the truth table is displayed.

Good luck and have fun!