Propositional Logic and Logic Programming
(Version: October 23, 2012)

1 Propositional Logic

Let set $\mathcal{A}$ be a set of **atomic propositions**. A world (usually denoted W) is simply a subset of $\mathcal{A}$. Intuitively, a atomic proposition (or "atom" for short) represents some attribute of reality. A world W tells us that all atoms in W are true and the rest are not.

The power set of worlds (the set of all subsets), denoted $2^{\mathcal{A}}$ is the set of possible worlds. Clearly, there are an exponential number of these.

We define a *formula* f over $\mathcal{A}$ using conjunction, disjunction, and negation $(\wedge, \vee, \neg)$ and inductively define a formula as follows:

- Each single $a \in \mathcal{A}$ is a formula.
- If f' is a formula, $\neg f'$ is also a formula
- If f', f'' are formulas, $f' \wedge f''$ is also a formula
- If f', f'' are formulas, $f' \vee f''$ is also a formula

Given a world W and formula f, we may ask the question "is f true in world W"? To deal with this issue, we define *satisfaction*, denoted $\models$. We say W *satisfies* f (written $W \models f$) if f is true in W. Likewise we will use $W \not\models f$ is W does not satisfy f. We define this relationship inductively as follows:

- If $f = a$, $W \models f$ iff $a \in W$
- If $f = \neg f'$ then $W \models f$ iff $W \not\models f$
- If $f = f' \wedge f''$ then $W \models f$ iff $W \models f'$ and $W \models f''$
- If $f = f' \vee f''$ then $W \models f$ iff $W \models f'$ or $W \models f''$

So, lets suppose we have an **agent** that collects information in the form of formulas (sometimes called sentences). Such a collection of formulas is often called a "program" or a "knowledge base" depending on what papers you read. Simply put, a program Π is a set of formulas that are known to describe reality. A world $W \models \Pi$ iff for all $f \in \Pi$ we have $W \models f$.

There are two problems that are primarily studied when dealing with programs. First is *consistency*. A program Π is consistent if there is a world W that satisfies it (hence satisfies all formulas in Π). An example of an **inconsistent** program would be something like $\{a, \neg a\}$ as clearly a cannot be both true and false at the same time. Determining if a program is consistent is an NP-hard problem.

The next problem is **entailment**. Given program Π (what we know about reality) and some formula f that we want to determine if we think that f can be concluded from the information in Π. We define this problem formally as follows. Let $M(\Pi)$ and $M(f)$ denote the sets of all worlds that satisfy Π and f respectively. We say that Π entails f (written $\Pi \models f$) iff $M(\Pi) \subseteq M(f)$. This problem is also very difficult, it is coNP-hard (the complexity class complementary to NP).

2 Simple Logic Programming

Noticeably absent from our notation on formulas is implication, $\rightarrow$. Given formulas f, f', we say world $W \models f \rightarrow f'$ iff W satisfies f is always true when W satisfies f'. This clearly can be derived using the previously defined Boolean operators: $f \rightarrow f' = \neg f \vee f'$. Logic programmers typically write the arrow going the other way, $\leftarrow$. Hence, $b \leftarrow a$ simply means a implies b. Such a construct is often called a *rule*. We will introduce a simple logical programming paradigm here where the rules are of the format $b \leftarrow a_1 \wedge a_2 \wedge \ldots \wedge a_n$ which simply means that if all of atoms $a_1, \ldots, a_n$ (called the "body") are true then atom b (called the "head") must also be true. These rules can be learned from historical data (i.e. a child touching a hot stove and getting burned), extracted from historical data (i.e. every email with a link to evilstuff.ru is malicious), or can come from an expert (i.e. a boxer with long arms throws a jab). In our language, we shall assume that our program Π consists of rules of this sort. Additionally, we shall also include a special atom TRUE where for all worlds W, $W \models \mathsf{TRUE}$. This allows us to have rules like $b \leftarrow \mathsf{TRUE}$ meaning that b is always true. This is called a **fact**. So, it should be obvious that checking the consistency of such a logic program can be achieved in polynomial time. This might be a good test question.

Lets talk about entailment. Suppose we want to see if a particular atom a is entails by Π (which consists of our rules). Well, there is always the brute-force approach. We could list out all possible worlds and check if they satisfy Π. Then, of the worlds that satisfy Π, we could see which ones contain a. If all of them satisfy f, then Π entails f. What is the time-complexity of this approach? How about space? Well, based on the correct answer to those two questions, you might realize that this is not the most efficient approach. After all, consistency is easy when we have these rules, why not entailment? So, lets introduce what is called a **fixed point operator**. For a given program, Π, we define the function $\mathbf{T}_\Pi : 2^W \rightarrow 2^W$. This function is defined as follows:

$$\mathbf{T}_\Pi(W) = W \cup \bigcup_{r \in \Pi} \{head(r) | body(r) \subseteq W\} \tag{1}$$

Where $head(r)$ is the atom in the head of rule r and $body(r)$ is the set of atoms in the body of r. So, next we can define multiple applications of $\mathbf{T}_\Pi$.

$$\mathbf{T}_\Pi^{(n)}(W) = \begin{cases} \mathbf{T}_\Pi(W) & \text{if } n = 1 \\ \mathbf{T}_\Pi(\mathbf{T}_\Pi^{(n-1)}(W)) & \text{otherwise} \end{cases}$$

So, let us suppose that there exists some natural number x such that $\mathbf{T}_\Pi^{(x-1)}(W) = \mathbf{T}_\Pi^{(x)}(W)$. If such a number exists, then we say that $\mathbf{T}_\Pi$ has a (least) **fixed point**. And guess what, we can prove that $\mathbf{T}_\Pi$ here *does* have a fixed point using Lattice Theory (which is beyond the scope of this course). So, now suppose we calculate the least fixed point of $\mathbf{T}_\Pi(\emptyset)$ (denoted $lfp\mathbf{T}_\Pi(\emptyset)$). This gives us a set of atoms. It can be proved that any world that satisfies Π is a superset of $lfp\mathbf{T}_\Pi(\emptyset)$. Hence, this set is called the **minimal model** of Π. So, if a (remember, this is the atom we are checking is entailed) is not in the minimal model of Π, then we know that Π does not entail a, as the minimal model is a world that is not in the set $M(a)$ yet satisfies Π. If a is in the minimal model, then we know that **all** worlds satisfying Π must contain a, hence we can be assured of entailment in this case. So, use the $\mathbf{T}_\Pi$ operator, can we

prove that entailment of this simple logical language can be done in polynomial time? Again, this might show up as a test question.