CSE 505: Class Notes on Semantics of Logic Programs

In this tutorial, we will concentrate only on propositional logic programs, i.e., logic programs where all predicates have arity 0. Hereafter, when we say logic programs, we will assume it is propositional unless otherwise stated.

Let P be a set of propositions. A literal is either p or $\neg p$ where $p \in P$ is a proposition. A logic program, denoted by $\mathcal{P}$, is a set of *clauses* represented as

$$p :- \alpha$$

where p is a proposition symbol, and α is a (possibly empty) comma-separated list of literals. We say p is the head of the clause and α is its body. When α is empty, the clause is simply represented as

$$p.$$

that is, as a fact.

1 Definite Logic Programs

A program $\mathcal{P}$ where no clause body contains a negative literal is called a *definite program*. In other words, if every clause in a program is a Horn clause, then it is a definite program.

The least model of a definite logic program $\mathcal{P}$ can be computed as the least fixed point of the *immediate consequence operator*, $T_{\mathcal{P}}$ of $\mathcal{P}$ defined as follows.

Definition 1 *Let $M \subseteq P$ be a set of propositions. Then $M' = T_{\mathcal{P}}(M)$, the immediate consequence of M is the least set such that:*

- $p \in M'$ *if* $p \in M$
- $p \in M'$ *if* $p :- q_1, q_2, \ldots q_n \in \mathcal{P}$ *and for all* $1 \leq i \leq n$, $q_i \in M$.

In other words, given the propositions that are true (in M) $T_{\mathcal{P}}$ computes the set of propositions whose truth follows from the program $\mathcal{P}$.

Example 1 *Consider the program $\mathcal{P}_1$:*

$$\begin{aligned} &p :- q, r. \\ &p :- s, t. \\ &r :- t. \\ &q. \\ &t. \end{aligned}$$

Let $M_0 = \{\}$. Then $T_{\mathcal{P}_1}(M_0) = \{q, t\}$. Now let $M_1 = \{q, t\}$. Then $T_{\mathcal{P}_1}(M_1) = \{q, r, t\}$. □

We will use the $T_{\mathcal{P}}$ operator to compute the least model of $\mathcal{P}$. Before we describe that result, we recall the following standard result from lattice theory.

Fixed point iterations over lattices Let $L = (S, \preceq)$ be a complete lattice, where S is some finite set and $\preceq$ is a partial order over S. Let $\perp$ be the least element in S according to the partial order $\preceq$. A function $f : S \to S$ is said to be monotonic over L if for all $x, y \in S$, $x \preceq y \Rightarrow f(x) \preceq f(y)$. An element x is a fixed point of function $f : S \to S$ if $f(x) = x$.

Let F be the set of all fixed points of f. Then the *least fixed point* of f, denoted by μf is an element $x \in F$ such that $\forall y \in F \; x \preceq y$.

Theorem 1 (Tarski-Knaster) *Given a lattice $(S, \preceq)$, the least fixed point of a monotonic function $f : S \to S$ is the limit of the sequence $x_0, x_1, \ldots, x_i, \ldots$ such that $x_0 = \perp$ and $x_{i+1} = f(x_i)$ for all $i \geq 0$.*

Proof sketch: We first show that the sequence $x_0, x_1, \ldots, x_i, \ldots$ converges, i.e., there is a $N \geq 0$ such that $x_{i+1} = x_i$ for all $i \geq N$. For that, we show that $x_i \preceq x_{i+1}$ for all $i \geq 0$ by induction on i. For the base case, note that $x_0 = \perp$ and hence $x_0 \preceq x_1$. Now, as induction hypothesis assume that $x_j \preceq x_{j+1}$ for some $j \geq 0$. By monotonicity of f, we have $f(x_j) \preceq f(x_{j+1})$; but $f(x_j) = x_{j+1}$ and $f(x_{j+1}) = x_{j+2}$ from construction of the sequence and hence $x_{j+1} \preceq x_{j+2}$, thereby completing the induction step.

Since S is finite, and $x_i \preceq x_{i+1}$ for all $i \geq 0$, there is some N for which $x_N = x_{N+1}$. It then follows from the definition of the sequence that $x_{i+1} = x_i$ for all $i \geq N$. The limit of the sequence is x_N, which is clearly a fixed point of f.

We now show that x_N is indeed the *least* fixed point. For that, consider some arbitrary fixed point y of f. We now show that $x_i \preceq y$ for all $i \geq 0$ by induction on i. For the base case, $\perp \preceq y$ and hence $x_0 \preceq y$. As induction hypothesis, assume that $x_j \preceq y$ for some j. By monotonicity of f, we have $f(x_j) \preceq f(y)$. Since y is a fixed point, and $f(x_j) = x_{j+1}$ by definition of the sequence, we have $x_{j+1} \preceq y$, thus completing the induction step. Hence it follows that the limit of the sequence $x_N \preceq y$, for any fixed point y, meaning that x_N is the least fixed point of f. □

Note: the original Tarski-Knaster theorem is more general and applies to potentially infinite complete partial orders; it is sufficient to restrict the result to finite lattices for our purposes.

Least model computation using $T_\mathcal{P}$ The immediate consequence operator maps a set of propositions to a set of propositions, and hence has the signature $T_\mathcal{P} : 2^P \to 2^P$ where 2^P represents the powerset of P. Note that $(2^P, \subseteq)$, the lattice formed by the subsets of P with subset ordering, is finite if P is finite.

We now consider the sequence $M_0, M_1, \ldots M_i, \ldots$ where $M_0 = \{\}$, the empty set and $\forall i \geq 0 \;\; M_{i+1} = T_\mathcal{P}(M_i)$. From definition of $T_\mathcal{P}$ it is easy to see that it is monotonic with respect to $\subseteq$ ordering. From the Tarski-Kanster theorem, it follows that the sequence of M_i's converge to the least fixed point of $T_\mathcal{P}$.

All that is left to show is that the least fixed point of $T_\mathcal{P}$ is also the least model of $\mathcal{P}$.

Theorem 2 *M is a model of program $\mathcal{P}$ if and only if M is a fixed point of $T_\mathcal{P}$.*

Proof sketch: To show that every model M of $\mathcal{P}$, is a fixed point of $\mathcal{P}$: assume to the contrary that there is some model M' which is not a fixed point of $T_\mathcal{P}$. From definition of $T_\mathcal{P}$, $M' \subseteq T_\mathcal{P}(M')$. Hence, there is some $p \in T_\mathcal{P}(M') - M'$. Since $p \in T_\mathcal{P}(M')$, there is some clause $p :- q_1, \ldots q_k$ such that $q_1, \ldots q_k \in M'$. If M' is a model of $\mathcal{P}$, p must be in M'. Thus every proposition in $T_\mathcal{P}(M')$ is also in M', and hence $T_\mathcal{P}(M') = M'$. □

Hence the least model of a program $\mathcal{P}$ is the least fixed point of $T_\mathcal{P}$.

2 Logic Programs with Stratified Negation

We now consider programs where the clause bodies may contain negative literals. Such programs are called normal logic programs. Below, we consider a subset of normal logic programs called stratified programs.

Given a program $\mathcal{P}$, we define its predicate dependency graph $G_{\mathcal{P}}$ as follows. The vertices of $G_{\mathcal{P}}$ are the propositions in $\mathcal{P}$. The edges in $G_{\mathcal{P}}$ are labelled as either positive or negative. There is a *positive* edge from p to r in $G_{\mathcal{P}}$ if and only if there is a clause of the form $p :- l_1, l_2, \ldots, l_k$ such that $l_i = r$ for some $i \leq k$. There is a *negative* edge from p to s in $G_{\mathcal{P}}$ if and only if there is a clause of the form $p :- l_1, l_2, \ldots, l_k$ such that $l_i = \neg s$ for some $i \leq k$.

Example 2 *Consider the program $\mathcal{P}_2$:*

$$p :- p, q.$$
$$q :- p.$$
$$r :- \neg s.$$
$$s :- \neg q.$$

The vertices of $G_{\mathcal{P}_2}$ are $\{p, q, r, s\}$. There are three positive edges in $G_{\mathcal{P}_2}$: namely, (p, p), (p, q) and (q, p). There are two negative edges, namely (r, s) and (s, q).

Example 3 *Consider the program $\mathcal{P}_3$:*

$$p :- \neg q.$$
$$q :- \neg r.$$
$$r :- p.$$

The vertices of $G_{\mathcal{P}_3}$ are $\{p, q, r\}$. There is one positive edge (r, p) and there are two negative edges (p, q) and (q, r).

There is a path from p to q in $G_{\mathcal{P}}$ iff there is a sequence of edges (positive or negative) by which we can reach q form p.

We say that a program $\mathcal{P}$ is *stratified* iff whenever there is a negative edge in $G_{\mathcal{P}}$ from q to p, there is *no* path from p to q. In other words, a program is stratified if negative edges in its predicate dependency graph do not participate in cycles.

Among the above examples, $\mathcal{P}_2$ is stratified. However $\mathcal{P}_3$ is not stratified, since there is a path from q to p via r, and there is also a negative edge from p to q.

The word "stratifed" stems from the observation that in such programs it is possible to partition the set of propositions into different "strata": sets $S_0, S_1, \ldots, S_k$ such that whenever $p \in S_i$ and there is a negative edge in $G_{\mathcal{P}}$ from p to q, then $q \in S_j$ for some $j < i$ (i.e. a strictly lower stratum).

Given such a stratification, we can partition the clauses in a program $\mathcal{P}$ into sets $\mathcal{P}^0, \mathcal{P}^1, \ldots \mathcal{P}^k$ such that $p :- \alpha \in \mathcal{P}^i$ iff $p :- \alpha \in \mathcal{P}$ and $p \in S_i$. Each of the sets $\mathcal{P}^i$ in the partition can itself be considered as a logic program. Note that the clauses in $\mathcal{P}^0$ form a definite logic program.

Example 4 *Consider the program $\mathcal{P}_2$ in Example 2. The set of propositions in $\mathcal{P}_2$ can be partitioned into strata $S_0 = \{p, q\}$, $S_1 = \{s\}$, and $S_2 = \{r\}$. The programs corresponding to these strata are $\mathcal{P}_2^0 = \{p :- p, q.\ q :- p\}$, $\mathcal{P}_2^1 = \{s :- \neg q\}$, and $\mathcal{P}_2^2 = \{r :- \neg s\}$.*

We now describe a computational procedure to evaluate what is called as the *perfect* model of a stratified logic program.

Definition 2 *For any set of propositions S, let $\mathcal{P}(S)$ be the program obtained from $\mathcal{P}$ by deleting*

1. *each clause that has a negative literal $\neg q$ in its body such that $q \in S$, and*
2. *all negative literals in the bodies of the remaining clauses, and*
3. *each positive literal q in the bodies of the remaining clauses such that $q \in S$.*

Intuitively, given a program $\mathcal{P}$ and set of propositions S, $\mathcal{P}(S)$ is the program obtained by "plugging in" the values of literals as prescribed by S. It is important to note that $\mathcal{P}(S)$ will always be a definite logic program.

Definition 3 *Let $\mathcal{P}$ be a stratified program, and $S_0, S_1, \ldots S_k$ be sets of predicates in the corresponding strata. Let $M_0, M_1, \ldots M_{k+1}$ be a sequence such that*

1. *$M_0 = \{\}$ and*
2. *$M_{i+1} = M_i \cup \mu T_{\mathcal{P}^i(M_i)}$ for all $i \leq k$.*

Then M_{k+1} is the perfect model of $\mathcal{P}$.

From the above definition it is clear that M_1 is the least model of the program $\mathcal{P}^0$. At each stratum, we "plug the values" of predicates defined in the lower strata and evaluate the least model of the resultant program. Note that $\mathcal{P}^i(M_i)$, the resultant program at stratum $i+1$ is a definite logic program. Consequently, the immediate consequence operator applied in each stratum is monotonic and hence its least fixed point is well defined.

Example 5 *The least model of $\mathcal{P}_2$ in Example 2 is evaluated as follows. (using the strata in Example 4).*

$$
\begin{aligned}
&M_0 = \{\}. \\
&M_1 = M_0 \cup \mu T_{\mathcal{P}^0(M_0)} = \mu T_{\mathcal{P}'} = \{\} \\
&M_2 = M_1 \cup \mu T_{\mathcal{P}^1(M_1)} = \mu T_{\{s.\}} = \{s\} \\
&M_3 = M_2 \cup \mu T_{\mathcal{P}^2(M_2)} = \{s\} \cup \mu T_{\{\}} = \{s\}
\end{aligned}
$$

It is easy to see that the perfect model of a definite program coincides with its least model.

3 Normal Logic Programs and Stable Model Semantics

While most logic *programs* that are written to express a given computation procedure are stratified, the same is not true for many logic programs that are used to represent knowledge. For instance, consider the following example, taken from Przymusinski's paper that appeared in Annals of Mathematics and Artificial Intelligence in 1994.

Example 6 *Consider the following program* $\mathcal{P}_5$*:*

$$\begin{array}{rcl} work & :- & \neg tired. \\ sleep & :- & \neg work. \\ tired & :- & sleep. \\ angry & :- & \neg paid, work. \\ paid. & & \end{array}$$

This program is non-stratified due to the cycles containing negation involving work, sleep and tired. However, since paid is a fact, angry is always false.

The semantics we have defined so far do not give a meaning to the above program, and hence cannot substantiate why *angry* should be false.

Stable models are a well-accepted semantics for non-stratified programs. They are computed based on a *quotient* operator defined as follows:

Definition 4 (Quotient) *For any set of propositions* S*, let* $\frac{\mathcal{P}}{S}$ *be the program obtained from* $\mathcal{P}$ *by deleting*

1. *each clause that has a negative literal* $\neg q$ *in its body such that* $q \in S$*, and*
2. *all negative literals in the bodies of the remaining clauses,*

The program $\frac{\mathcal{P}}{S}$ *is called the* quotient *of* $\mathcal{P}$ *with respect to* S*.*

The operation for obtaining the quotient of a program with respect to a set of propositions is called the Gelfond-Lifschitz transformation. Note the similarity between the "plug-in" operator $\mathcal{P}(S)$ defined earlier for stratified programs, and the quotient operator above: the quotient operator "plugs-in" th evalues only for negative literals. Note that the quotient is always a definite logic program.

Example 7 *Consider* $S_1 = \{work\}$ *and the program* $\mathcal{P}_5$ *from the previous example. The quotient* $\frac{\mathcal{P}_5}{S_1}$ *is*

$$\begin{array}{rcl} work. & & \\ tired & :- & sleep. \\ angry & :- & work. \\ paid. & & \end{array}$$

Let $S_2 = \{paid\}$*. The quotient* $\frac{\mathcal{P}_5}{S_2}$ *is*

$$\begin{array}{rcl} work. & & \\ sleep. & & \\ tired & :- & sleep. \\ paid. & & \end{array}$$

Let $S_3 = \{work, paid\}$*. The quotient* $\frac{\mathcal{P}_5}{S_3}$ *is*

$$\begin{array}{rcl} work. & & \\ tired & :- & sleep. \\ paid. & & \end{array}$$

Definition 5 (Stable Model) *A set of predicates M is a stable model of $\mathcal{P}$ if and only if* $M = \mu T_{\frac{\mathcal{P}}{M}}$.

In other words, a set of predicates is a stable model iff it is also the least model of the quotient program. For example, the set $\{work\}$ is not a stable model of $\mathcal{P}_5$ since the least model of its corresponding quotient is $\{work, angry, paid\}$. On the other hand, the set $S_3 = \{work, paid\}$ is a stable model since the least model of $\frac{\mathcal{P}_5}{S_3}$ is $\{work, paid\}$.

Programs may not always have stable models. For instance the program $\{p :- \neg p.\}$ has no stable model. Programs may have more than one stable model. For instance, program P_5 has another stable model $\{tired, sleep, paid\}$.

Every stratified program has a unique stable model which coincides with its perfect model. Consequently, the least model of a definite logic program coincides with its unique stable model.

For more on Stable Models, see the paper by Gelfond and Lifschitz in the "Online Resources" page of the course web site.