Lecture 2: Positive Programs

Outline
1. Background for rules
2. Rules and programs
3. Minimal models
4. Constructing the least model
5. Programs with variables
6. Expressive power

1. BACKGROUND FOR RULES

- Answer set programming adopts a rule-based syntax previously used in PROLOG, deductive databases, and expert systems.
- **Horn clauses** provide rule-based reasoning with a solid foundation:
  1. Rules can be interpreted as Horn clauses.
  2. Classical models determine the set of logical consequences $\text{Cn}(R)$ associated with each set of rules $R$.
- Horn clauses lend themselves for efficient implementation which makes them important from the computational point of view.
- Typically, applications require a more expressive language but for now we concentrate on rules corresponding to Horn clauses.

Definitions.
1. A **literal** is either an atom $a$ (a **positive literal**) or the negation of an atom $\neg a$ (a **negative literal**).
2. A **clause** is a disjunction $l_1 \lor \ldots \lor l_n$ of literals $l_1, \ldots, l_n$.
3. A **Horn clause** is a clause with at most one positive literal.
4. A **program clause**, or a **rule** for short, is a disjunction of literals $a \lor \neg b_1 \lor \ldots \lor \neg b_n$ with exactly one positive literal.

Example. The clauses $\neg p \lor \neg q \lor \neg r$ and $\neg p \lor q \lor \neg r$ are Horn clauses but $p \lor \neg q \lor r$ is not. Only the second one is a program clause.

2. RULES AND PROGRAMS

Certain notational conventions are adopted for Horn clauses:
- A **rule** $a \lor \neg b_1 \lor \ldots \lor \neg b_n$ is written $a \leftarrow b_1, \ldots, b_n$ where $a$ and $b_1, \ldots, b_n$ form the **head** and the **body** of the rule, respectively.
- A **constraint** $\neg b_1 \lor \ldots \lor \neg b_n$ is written $\leftarrow b_1, \ldots, b_n$ and it can be viewed as a rule with an empty head.
- A **fact** $a$ ($n = 0$) is a rule written without "\leftarrow".
- Full stops "." are also used to separate rules in sets of rules.

Definition. A **positive program** $P$ is a set of rules as defined above.

Remark. Here the word “positive” refers to the fact that rule bodies are negation-free. Forms of negation will be introduced later.
Satisfaction and Entailment

Definitions. Assume rules and constraints based on a set of atoms \( \mathcal{P} \).

1. A rule \( a \leftarrow b_1, \ldots, b_n \) is satisfied in an interpretation \( I \subseteq \mathcal{P} \), denoted \( I \models a \leftarrow b_1, \ldots, b_n \), iff \( \{b_1, \ldots, b_n\} \subseteq I \) implies \( a \in I \).

2. A constraint \( \leftarrow b_1, \ldots, b_n \) is satisfied in an interpretation \( I \subseteq \mathcal{P} \), denoted \( I \models \leftarrow b_1, \ldots, b_n \), iff \( \{b_1, \ldots, b_n\} \not\subseteq I \).

3. An interpretation \( I \subseteq \mathcal{P} \) is a model of a set of rules and constraints \( P \cup C \), denoted \( M \models P \cup C \), iff \( M \models r \) for each \( r \in P \cup C \).

4. An atom \( a \) is a logical consequence of \( P \cup C \) iff \( a \in M \) for every interpretation \( M \subseteq \mathcal{P} \) such that \( M \models P \cup C \).

Proposition. Every positive program \( P \) is satisfiable (has a model).

Proof. The interpretation \( M = \text{Hb}(P) \) is trivially a model of \( P \). \( \square \)

3. Minimal Models

Definition.

1. An interpretation \( M \subseteq \mathcal{P} \), represented as the set of atoms true in \( M \), is smaller than another interpretation \( N \subseteq \mathcal{P} \) iff \( M \subset N \).

2. An interpretation \( M \subseteq \text{Hb}(P) \) is a minimal model of a (positive) program \( P \) iff \( M \models P \) and there is no smaller model \( N \models P \).

Example. Consider the following positive program:

\[
P = \{ q \leftarrow r, \ r \leftarrow p, q \}.
\]

- The interpretation \( M = \{ q, r \} \) is a model of \( P \).
- However, \( M \) is not minimal because \( N = \emptyset \) is also a model of \( P \).
- But, in contrast, \( N \) is a minimal model of \( P \).

Translating Constraints into Rules

- Any set Horn clauses \( S \), effectively a union \( P \cup C \) of a positive program \( P \) and a set of constraints \( C \), is not satisfiable in general.
- E.g., \( \{ p, \neg p \} \) corresponding to \( \{ p \} \cup \{ \leftarrow p \} \) has no models.
- Any set of constraints \( C \) can be translated into a positive program \( \text{Tr}_{\text{RULE}}(C) = \{ \perp \leftarrow b_1, \ldots, b_n | \leftarrow b_1, \ldots, b_n \in C \} \)
  where \( \perp \) is a new atom not appearing in \( \text{Hb}(P) \) nor \( \text{Hb}(C) \).

Proposition. A set of Horn clauses \( S \), viewed as a union \( P \cup C \) in the way explained above, is satisfiable \( \iff P \cup \text{Tr}_{\text{RULE}}(C) \not\models \perp \).

Example. The unsatisfiability of \( \{ p, \neg p \} \) can be determined using the translation given above: \( \{ p \} \cup \text{Tr}_{\text{RULE}}(\{ \leftarrow p \}) = \{ p, \perp \leftarrow p \} \models \perp. \)

Properties of Minimal Models (I)

Theorem. If \( M_i \subseteq \text{Hb}(P) \) (where \( i \in I \)) is a collection of models for a positive program \( P \), then \( M = \bigcap \{ M_i | i \in I \} \) is also a model of \( P \).

Proof. Suppose that \( M \not\models P \).

\[
\Rightarrow \exists a \leftarrow b_1, \ldots, b_n \in P \text{ such that } \{b_1, \ldots, b_n\} \subseteq M \text{ but } a \notin M \\
\Rightarrow \{b_1, \ldots, b_n\} \subseteq M_i \text{ for all } i \in I \\
\Rightarrow a \in M_i \text{ for all } i \in I \text{ because } M_i \models P, \\
a \leftarrow b_1, \ldots, b_n \in P \text{ and } M_i \models a \leftarrow b_1, \ldots, b_n \\
\Rightarrow a \in M = \bigcap \{ M_i | i \in I \}, \text{ a contradiction.}
\]

Thus \( M \models P \) is necessarily the case. \( \square \)
Properties of Minimal Models (II)

**Theorem.** A positive program $P$ has at least one minimal model.

**Proof.** We will cover the case when $|\text{Hb}(P)| = n$ is finite (a generalization for the infinite case requires transfinite induction).

Since $P$ is a positive program, we know that $M_0 = \text{Hb}(P) \models P$. Then define a decreasing sequence $M_0 \supseteq \cdots \supseteq M_i \supseteq \cdots$ of models for $P$.

- If $M_i$ is a minimal model of $P$, let $M_{i+1} = M_i$.
- If $M_i$ is not a minimal model of $P$, it has a model $N \subseteq M_i$.

Let $M_{i+1} = N$.

Assuming that $M_i \models P$ is never minimal implies that the sequence is properly decreasing for all $i \geq 0$. A contradiction when $i > n$.

Example

The idea of the preceding proof can be demonstrated using

$$P_n = \{ p_0 \leftarrow p_0. \; p_1 \leftarrow p_1. \; p_2 \leftarrow p_2. \; \cdots \; p_n \leftarrow p_n. \}$$

which is a positive program with a finite number, i.e., $n+1$, of rules.

- The interpretation $M_0 = \text{Hb}(P_n) = \{ p_0, \ldots, p_n \}$ is a model of $P$ but not minimal because $M_1 = \{ p_1, \ldots, p_n \} \models P$.

- A generalization for $i > 0$: the interpretation $M_i = \{ p_i, \ldots, p_n \}$ is a model of $P_n$ but not minimal because $M_{i+1} = \{ p_{i+1}, \ldots, p_n \} \models P_n$.

- When $i$ equals to $|\text{Hb}(P_n)| = n+1$, we have a minimal model $M = \bigcap_{i=0}^{n+1} M_i = \emptyset$.

Properties of Minimal Models (III)

**Theorem.** Every positive program $P$ has a unique minimal model, the least model $\text{LM}(P)$ of $P$, which is the intersection of its all models.

**Proof.** Since $P$ has at least one minimal model (shown above), let us assume that $P$ had two minimal models, say $M_1$ and $M_2$.

\[
\Rightarrow M_1 \cap M_2 \models P \\
\Rightarrow M_1 \cap M_2 = M_1 \text{ and } M_1 \cap M_2 = M_2 \text{ (M_1 and M_2 are minimal)} \\
\Rightarrow M_1 = M_2.
\]

Thus $\text{LM}(P) \subseteq M$ holds for every $M \models P$ because $\text{LM}(P)$ is unique.

Since $\text{LM}(P) \models P$, we obtain $\bigcap \{ M \subseteq \text{Hb}(P) \mid M \models P \} = \text{LM}(P)$.

Answer Sets

**Corollary.** For any positive program $P$,

$$\text{LM}(P) = \{ a \in \text{Hb}(P) \mid P \models a \}.$$ 

- By this corollary, the least model of a positive program $P$ provides means to answer queries about atoms in $\text{Hb}(P)$.
- Thus $\text{LM}(P)$ is the unique answer set associated with $P$.

**Example.** For $P = \{ a \leftarrow b, c. \; b \leftarrow a, c. \; c \leftarrow a, b. \}$,

1. $\text{LM}(P \cup \{ a. \}) = \{ a \}$ and
2. $\text{LM}(P \cup \{ a, b. \}) = \{ a, b, c \}$.

Thus $P \cup \{ a. \} \not\models c$ but $P \cup \{ a, b. \} \models c$. 
4. CONSTRUCTING THE LEAST MODEL

**Definition.** Let \( P \) be a positive logic program. Then define an operator \( T_P : 2^{Hb(P)} \rightarrow 2^{Hb(P)} \) on interpretations \( I \subseteq Hb(P) \) as follows:

\[
T_P(I) = \{ a \in Hb(P) \mid a \leftarrow b_1, \ldots, b_n \in P \text{ and } \{ b_1, \ldots, b_n \} \subseteq I \}.
\]

An interpretation \( I \) is a **fixpoint** of the operator \( T_P \) iff \( T_P(I) = I \).

A fixpoint \( I \) is the **least fixpoint** of \( T_P \) iff \( I \subseteq I' \) for every \( I' = T_P(I') \).

**Example.** Let us analyze \( P = \{ a \leftarrow a, b \leftarrow b, c \leftarrow b, d \leftarrow a, b \} \).

1. Now \( T_P(\{a\}) = \{a, b\} \) and \( T_P(\{a, b\}) = \{a, b, c, d\} \).
2. The interpretation \( M_1 = \{a, b, c, d\} \) is a fixpoint of \( T_P \) since \( T_P(M_1) = \{a, b, c, d\} = M_1 \), and
3. The interpretation \( M_2 = \{b, c\} \) is the least fixpoint of \( T_P \).

© 2007 TKK / TCS

**Properties of \( T_P \)**

**Proposition.** An interpretation \( M \subseteq Hb(P) \) is a model of a positive program \( P \) iff \( T_P(M) \subseteq M \).

**Proof.** For any interpretation \( M \subseteq Hb(P) \), \( M \not\models P \)

\[
\Leftrightarrow \exists a \leftarrow b_1, \ldots, b_n \in P \text{ such that } \{ b_1, \ldots, b_n \} \subseteq M \text{ but } a \not\in M
\]

\[
\Leftrightarrow \exists a \in T_P(M) \text{ such that } a \not\in M \Leftrightarrow T_P(M) \not\subseteq M.
\]

© 2007 TKK / TCS

**Proposition.** (Monotonicity) For a positive program \( P \),

\[
M \subseteq N \subseteq Hb(P) \text{ implies } T_P(M) \subseteq T_P(N).
\]

**Proof.** For any atom \( a \in Hb(P) \), we have that \( a \in T_P(M) \)

\[
\Leftrightarrow \exists a \leftarrow b_1, \ldots, b_n \in P \text{ such that } \{ b_1, \ldots, b_n \} \subseteq M
\]

\[
\Leftrightarrow \exists a \leftarrow b_1, \ldots, b_n \in P \text{ such that } \{ b_1, \ldots, b_n \} \subseteq N \text{ (} M \subseteq N \)
\]

\[
\Rightarrow a \in T_P(N).
\]

© 2007 TKK / TCS

**Properties of the Least Fixpoint (I)**

**Proposition.** For a positive program \( P \), the operator \( T_P \) has the least fixpoint \( \operatorname{lfp}(T_P) = \bigcap\{ M \subseteq Hb(P) \mid M = T_P(M) \} \).

**Proof.** Every monotonic operator has a least fixpoint (Knaster-Tarski) which is unique. For \( T_P \), we denote this fixpoint by \( \operatorname{lfp}(T_P) \).

For the intersection property, it is sufficient to note that by definition \( \operatorname{lfp}(T_P) \subseteq M \) for any \( M = T_P(M) \), and \( M = \operatorname{lfp}(T_P) \) in particular. \( \square \)

The unique fixpoint \( \operatorname{lfp}(T_P) \) can be constructed iteratively:

**Definition.** For a positive program \( P \), define a sequence of interpretations by setting \( T_P \uparrow 0 = \emptyset \), \( T_P \uparrow i + 1 = T_P(T_P \uparrow i) \) for \( i > 0 \), and the **limit** \( T_P \uparrow \infty = \bigcup_{i=0}^\infty T_P \uparrow i \) of the sequence.

© 2007 TKK / TCS

**Properties of the Least Fixpoint (II)**

**Theorem.** For a positive program \( P \), \( \operatorname{lfp}(T_P) = T_P \uparrow \infty = \operatorname{LM}(P) \).

**Proof.** We will prove the claim \( \operatorname{lfp}(T_P) = T_P \uparrow \infty \) when \( P \) is **finite**; the infinite case uses **transfinite induction** and the **compactness** of \( T_P \).

(\( \subseteq \) ) The monotonicity of \( T_P \) guarantees that the sequence of interpretations \( T_P \uparrow i \) is increasing. Hence \( T_P(T_P \uparrow i) = T_P \uparrow i \) for some \( i \geq 0 \). Thus \( \operatorname{lfp}(T_P) \subseteq T_P \uparrow i \subseteq T_P \uparrow \infty \).

(\( \supseteq \) ) It follows by induction on \( i \) that \( T_P \uparrow i \subseteq \operatorname{lfp}(T_P) \) for every \( i \geq 0 \).

For \( \operatorname{lfp}(T_P) = \operatorname{LM}(P) \), we note the following:

(\( \subseteq \) ) It follows by induction that \( T_P \uparrow i \subseteq \operatorname{LM}(P) \) for every \( i \geq 0 \).

(\( \supseteq \) ) Any fixpoint \( M = T_P(M) \) is also a model of \( P \). Thus the intersection of models, i.e. \( \operatorname{LM}(P) \), is contained in \( M \). \( \square \)

© 2007 TKK / TCS
5. PROGRAMS WITH VARIABLES

- Disregarding any non-logical features, logic programs and deductive databases can be viewed as sets of rules of the form
  \[ P(t) \leftarrow P_1(t_1), \ldots, P_n(t_n) \]
  where \( P(t) \) and \( P_i(t_i) \)'s are atomic formulas involving lists of terms \( t, t_1, \ldots, t_n \) as their arguments.
- Variables appearing in rules are universally quantified.
- Each set of rules \( P \), also called a positive program in the sequel, has a Herbrand base \( \text{Hb}(P) \) associated with it.
- A rule \( P(t) \leftarrow P_1(t_1), \ldots, P_n(t_n) \) with variables \( x_1, \ldots, x_m \) stands for its all ground instances, each of which is obtained by substituting the variables \( x_1, \ldots, x_m \) by some ground terms \( s_1, \ldots, s_m \).
6. EXPRESSIVE POWER

Rules are expressive enough to cover basic operations on relations as present in relational algebra (SQL):

1. Union: EUNational(x) ← Finn(x).
   EUNational(x) ← Swede(x).
2. Intersection: Father(x) ← Parent(x), Man(x).
4. Selection: Millionaire(x) ← Assets(x,y), Greater(y, 999999).
5. Composition: Result(x,y) ← Student(x,i), Grade(i,y).

Contrast with Relational Algebra

Unlike SQL (stands for Structured Query Language), positive programs enable recursive definitions.

Example. E.g., the transitive closure of a relation is expressible:

Connection(x,y) ← Flight(x,y).
Connection(x,y) ← Flight(x,z), Connection(z,y).

On the other hand, the conditions used in the form of rules considered so far cannot refer to complements of relations as in SQL.

Example. However, it is not trivial to add negation (∼ below):

Man(a), Man(b), Man(c).
Shaves(c,x) ← Man(x), ∼Shaves(x,x). Shaves(a,a).

OBJECTIVES

- You are able to define minimal models and the least model for positive programs and to prove simple properties about them.
- You know the interconnection between the least model of a positive program and its logical consequences.
- You are able to construct the least model for the given positive program $P$ by calculating the least fixpoint of $T_P$.
- You have some preliminary ideas how minimal models are exploited in knowledge representation.
- You are aware of the basic similarities and differences of relational algebra and rule-based languages.

TIME TO PONDER

Consider two positive programs $P_1$ and $P_2$ and their union $P_1 \cup P_2$.

Which of the following do hold in general?

1. $\text{LM}(P_1 \cup P_2) \subseteq \text{LM}(P_1) \cup \text{LM}(P_2)$.
2. $\text{LM}(P_1) \cup \text{LM}(P_2) \subseteq \text{LM}(P_1 \cup P_2)$.