Lecture 5: Deductive Databases

2-AIN-144/2-IKV-131 Knowledge Reperesentation & Reasoning

Martin Baláž, Martin Homola

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

14 Mar 2013

Outline

Example: Logic Program without Negation

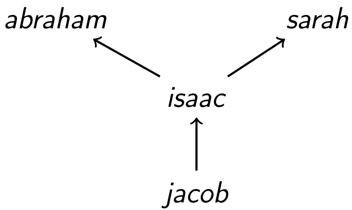
Extensional Database (EDB):

$parent(X, Y)$ X is a parent of Y

Intensional Database (IDB):

$ancestor(X, Y)$ X is an acestor of Y

Example: Logic Program without Negation



Note: $p \rightarrow q$ means q is a parent of p

$$
\begin{aligned}
parent(abraham, isaac) &\leftarrow \\
parent(sarah, isaac) &\leftarrow \\
parent(isaac, jacob) &\leftarrow
\end{aligned}
$$

$$
\begin{aligned}
ancestor(X, Y) &\leftarrow parent(X, Y) \\
ancestor(X, Y) &\leftarrow ancestor(X, Z), ancestor(Z, Y)
\end{aligned}
$$

The Language of Logic Programs

A *term* is

- a *variable* X
- a *function term* $f(t_1, t_2, \ldots, t_n)$ where f is a function symbol with arity n and $t_1, t_2, \ldots, t_n$ are terms.

An *atom* is a formula $p(t_1, t_2, \ldots, t_n)$ where p is a predicate symbol with arity n and $t_1, t_2, \ldots, t_n$ are terms.

A *literal* is an atom A (*positive literal*) or a negated atom *not* A (*negative literal*).

Definite Logic Program

Definition (Definite Logic Program)

A *definite logic program* is a set of rules

$$A_0 \leftarrow A_1, \ldots, A_m$$

where $0 \leq m$ and each A_i, $0 \leq i \leq m$, is an atom.

The *head* of a rule r is the atom $head(r) = A_0$ and the *body* of a rule r is the set of atoms $body(r) = \{A_1, \ldots, A_m\}$.

Rules with the empty body are called *facts*.

Logic Program as First-Order Theory

Each definite logic program can be viewed as a first-order theory.

Logic program P:

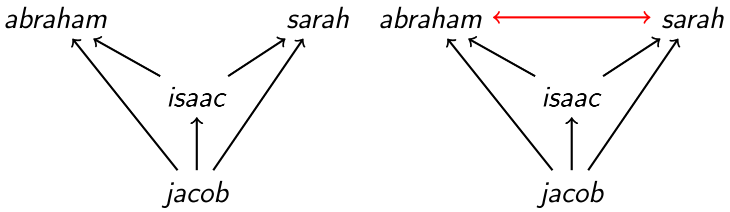
$$\begin{aligned} parent(abraham, isaac) &\leftarrow \\ parent(sarah, isaac) &\leftarrow \\ parent(isaac, jacob) &\leftarrow \\ ancestor(X, Y) &\leftarrow parent(X, Y) \\ ancestor(X, Y) &\leftarrow ancestor(X, Z), ancestor(Z, Y) \end{aligned}$$

First-order theory T:

$$parent(abraham, isaac) \quad parent(sarah, isaac) \quad parent(isaac, jacob)$$
$$\forall X \forall Y (parent(X, Y) \Rightarrow ancestor(X, Y))$$
$$\forall X \forall Y \forall Z (ancestor(X, Z), ancestor(Z, Y) \Rightarrow ancestor(X, Y))$$

Minimal Model as Intuitive Meaning

Not all models of T are intuitive.



Intuitive model

Unintuitive model

Note: $p \to q$ means q is an ancestor of p

Why Minimal Model?

Closed World Assumption

- We have complete knowledge about the world
- Usually there exist more negative facts then positive
- Therefore we provide only positive information and what is not known to be true is false

Open World Assumption

- We don't have complete knowledge about the world
- Usually the amount of positive information is comparable with the amount of negative information
- What is not known to be true or false is unknown

In databases, we usually assume closed world.

Bottom-Up Evaluation

Definition (Immediate Consequence Operator)

Let P be a definite logic program. The *immediate consequence operator* T_P is defined as follows:

$$T_P(I) = \{A \in \mathcal{B}_P \mid \exists r \in P : head(r) = A, I \models body(r)\}$$

The iteration of T_P is defined as follows:

$$\begin{aligned} T_P \uparrow 0(I) &= I \\ T_P \uparrow n+1(I) &= T_P(T_P \uparrow n(I)) \\ T_P \uparrow \omega(I) &= \bigcup_{n<\omega} T_P \uparrow n(I) \end{aligned}$$

Example: Logic Program without Negation

$$\begin{aligned}
parent(abraham, isaac) &\leftarrow \\
parent(sarah, isaac) &\leftarrow \\
parent(isaac, jacob) &\leftarrow \\
ancestor(X, Y) &\leftarrow parent(X, Y) \\
ancestor(X, Y) &\leftarrow ancestor(X, Z), ancestor(Z, Y)
\end{aligned}$$

$$\begin{aligned}
M_0 &= \emptyset \\
M_1 &= M_0 \cup \{parent(abraham, isaac), parent(sarah, isaac), \\
&\quad parent(isaac, jacob)\} \\
M_2 &= M_1 \cup \{ancestor(abraham, isaac), ancestor(sarah, isaac), \\
&\quad ancestor(isaac, jacob)\} \\
M_3 &= M_2 \cup \{ancestor(abraham, jacob), ancestor(sarah, jacob)\} \\
M_4 &= M_3
\end{aligned}$$

Model-Theoretic Semantics vs. Fixpoint Semantics

Proposition

Let P be a definite logic program. Then $\{A \in \mathcal{B}_P \mid P \models A\}$ is the least model of P.

Proposition

Let P be a definite logic program. Then $T_P \uparrow \omega(\emptyset)$ is the least model of P.

Model-theoretic semantics and fixpoint semantics coincide.

Example: Logic Program with Negation

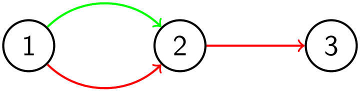
Extensional Database:

$red(X, Y)$	Red bus line runs from X to Y
$green(X, Y)$	Green bus line runs from X to Y

Intentional Database:

$greenPath(X, Y)$	You can get from X to Y using only green busses
$redMonopoly(X, Y)$	Red bus line runs from X to Y, but you can't get from X to Y using only green busses

Example: Logic Program with Negation



$$
\begin{aligned}
red(1,2) &\leftarrow \\
red(2,3) &\leftarrow \\
green(1,2) &\leftarrow
\end{aligned}
$$

$$
\begin{aligned}
greenPath(X,Y) &\leftarrow green(X,Y) \\
greenPath(X,Y) &\leftarrow greenPath(X,Z), greenPath(Z,Y) \\
redMonopoly(X,Y) &\leftarrow red(X,Y), not\ greenPath(X,Y)
\end{aligned}
$$

Normal Logic Program

Definition (Normal Logic Program)

A *normal logic program* is a set of rules

$$A_0 \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n$$

where $0 \leq m \leq n$ and each A_i, $0 \leq i \leq n$, is an atom.

The *head* of a rule r is the atom $head(r) = A_0$ and the *body* of a rule r is the set of literals $body(r) = \{A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n\}$.

Rules with the empty body are called *facts*.

What is the Problem with Negation?

Logic program P:

$$\begin{aligned} red(1,2) &\leftarrow \\ red(2,3) &\leftarrow \\ green(1,2) &\leftarrow \\ greenPath(X,Y) &\leftarrow green(X,Y) \\ greenPath(X,Y) &\leftarrow greenPath(X,Z), greenPath(Z,Y) \\ redMonopoly(X,Y) &\leftarrow red(X,Y), not\ greenPath(X,Y) \end{aligned}$$

We have two minimal models:

$$\begin{aligned} M_1 &= EDB \cup \{greenPath(1,2), redMonopoly(2,3)\} \\ M_2 &= EDB \cup \{greenPath(1,2), greenPath(2,3), greenPath(1,3)\} \end{aligned}$$

Only M_1 is the intuitive meaning of P!

Semi-Positive Logic Program

Definition (Semi-Positive Logic Program)

A normal logic program is *semi-positive* iff the only negated literals are literals from EDB.

Extensional Database:

$$\begin{aligned} red(1,2) &\leftarrow \\ red(2,3) &\leftarrow \\ green(1,2) &\leftarrow \end{aligned}$$

Intensional Database:

$$onlyRed(X,Y) \leftarrow red(X,Y), not\ green(X,Y)$$

Semi-Positive Model

Semi-Positive Model:

$$M = T_P \uparrow \omega(\emptyset)$$

$$\begin{aligned}
red(1,2) &\leftarrow \\
red(2,3) &\leftarrow \\
green(1,2) &\leftarrow \\
onlyRed(1,2) &\leftarrow red(1,2), not\ green(1,2) \\
onlyRed(2,3) &\leftarrow red(2,3), not\ green(2,3)
\end{aligned}$$

$$\begin{aligned}
M_0 &= \emptyset \\
M_1 &= T_P(M_0) = M_0 \cup \{red(1,2), red(2,3), green(1,2)\} \\
M_2 &= T_P(M_1) = M_1 \cup \{onlyRed(2,3)\} \\
M_3 &= T_P(M_2) = M_2
\end{aligned}$$

Logic Program which is not Semi-Positive

Extensional Database:

$$
\begin{aligned}
red(1,2) &\leftarrow \\
red(2,3) &\leftarrow \\
green(1,2) &\leftarrow
\end{aligned}
$$

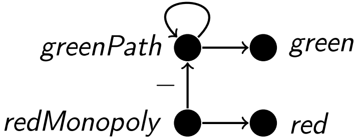
Intensional Database:

$$
\begin{aligned}
greenPath(X,Y) &\leftarrow green(X,Y) \\
greenPath(X,Y) &\leftarrow greenPath(X,Z), greenPath(Z,Y) \\
redMonopoly(X,Y) &\leftarrow red(X,Y), not\ greenPath(X,Y)
\end{aligned}
$$

The logic program is not semi-positive. The atom $redMonopoly(X,Y)$ depends on the literal $not\ greenPath(X,Y)$, but $greenPath(X,Y)$ is not from the extensional database.

Stratified Logic Program

$$
\begin{array}{rcl}
greenPath(X, Y) & \leftarrow & green(X, Y) \\
greenPath(X, Y) & \leftarrow & greenPath(X, Z), greenPath(Z, Y) \\
redMonopoly(X, Y) & \leftarrow & red(X, Y), not\ greenPath(X, Y)
\end{array}
$$

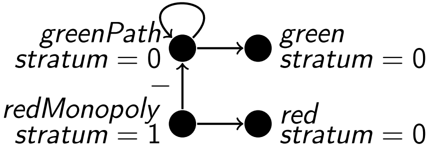


Dependency graph

- nodes are predicate symbols
- node p is connected to node q iff there is a rule which contains an atom with predicate symbol p in the head and a literal with predicate symbol q in the body
- an arc $p \rightarrow q$ is labeled $-$ if the literal containing q is negative

Stratified Logic Program

$$\begin{array}{rcl} greenPath(X,Y) & \leftarrow & green(X,Y) \\ greenPath(X,Y) & \leftarrow & greenPath(X,Z), greenPath(Z,Y) \\ redMonopoly(X,Y) & \leftarrow & red(X,Y), not\ greenPath(X,Y) \end{array}$$



The *stratum* of a predicate symbol is the largest number of negative edges on a path from that symbol.

A normal logic program is *stratified* if all predicte symbols have finite strata, otherwise it is *unstratified*.

Stratified Model

Stratified Logic Program:

$$P = P_0 \cup P_1 \cup \cdots \cup P_n$$

Each rule in P_i has in the head a predicate symbol with the stratum i.

Progressive Immediate Consequence Operator:

$$T^*_P(I) = T_P(I) \cup I$$

Stratified Model:

$$\begin{array}{rcl} M_0 & = & T^*_{P_0} \uparrow \omega(\emptyset) \\ M_1 & = & T^*_{P_1} \uparrow \omega(M_0) \\ & \vdots & \\ M_n & = & T^*_{P_n} \uparrow \omega(M_{n-1}) \end{array}$$

Example

$$P_0 = \left\{ \begin{array}{rcl} red(1,2) & \leftarrow & \\ red(2,3) & \leftarrow & \\ green(1,2) & \leftarrow & \\ greenPath(X,Y) & \leftarrow & green(X,Y) \\ greenPath(X,Y) & \leftarrow & greenPath(X,Z), \\ & & greenPath(Z,Y) \end{array} \right\}$$

$$P_1 = \left\{ \begin{array}{rcl} redMonopoly(X,Y) & \leftarrow & red(X,Y), \\ & & not\ greenPath(X,Y) \end{array} \right\}$$

$$\begin{array}{rcl} M_0 & = & \{red(1,2), red(2,3), green(1,2), greenPath(1,2)\} \\ M_1 & = & \{redMonopoly(2,3)\} \cup M_0 \end{array}$$

Properties of Stratified Logic Programs

Proposition

The stratified model of a normal logic program is minimal.

Proposition

A semi-positive logic program is stratified.

Proposition

Let P be a definite logic program. The stratified model of P coincides with the least model of P.

Unstratified Logic Program

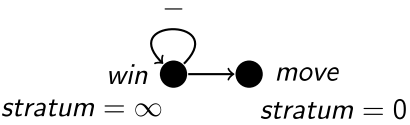
Extensional Database:

$$
\begin{aligned}
move(1,2) &\leftarrow \\
move(2,3) &\leftarrow \\
move(1,3) &\leftarrow
\end{aligned}
$$

Intentional Database:

$$win(X) \leftarrow move(X,Y), not\ win(Y)$$

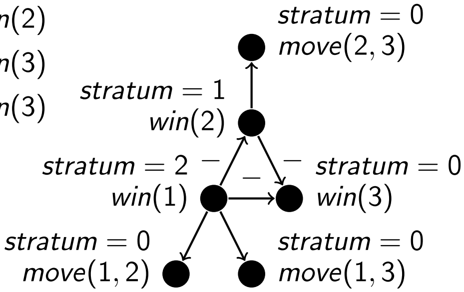
The logic program is unstratified:



Locally Stratified Logic Program

The following logic program is locally stratified:

$$move(1,2) \leftarrow$$
$$move(2,3) \leftarrow$$
$$move(1,3) \leftarrow$$
$$win(1) \leftarrow move(1,2), not\ win(2)$$
$$win(2) \leftarrow move(2,3), not\ win(3)$$
$$win(1) \leftarrow move(1,3), not\ win(3)$$



Locally Stratified Logic Program

Dependency graph

- nodes are ground atoms
- node p is connected to node q iff there exists a rule which contains p in the head and q in the body.
- an arc $p \to q$ is labeled $-$ if q occurs negative

The stratum of a ground atom is the largest number of negative edges on a path from that ground atom.

A normal logic program is *locally stratified* iff all ground atoms have finite strata, otherwise it is *locally unstratified*.

Example

$$P_0 = \left\{ \begin{array}{ll} move(1,2) & \leftarrow \\ move(2,3) & \leftarrow \\ move(1,3) & \leftarrow \end{array} \right\}$$

$$P_1 = \{\ win(2) \leftarrow move(2,3), not\ win(3)\ \}$$

$$P_2 = \left\{ \begin{array}{lll} win(1) & \leftarrow & move(1,2), not\ win(2) \\ win(1) & \leftarrow & move(1,3), not\ win(3) \end{array} \right\}$$

$$\begin{aligned} M_0 &= \{move(1,2), move(2,3), move(1,3)\} \\ M_1 &= \{win(2)\} \cup M_0 \\ M_2 &= \{win(1)\} \cup M_1 \end{aligned}$$

Properties of Locally Stratified Logic Programs

Proposition

The locally stratified model is minimal.

Proposition

A stratified logic program is locally stratified.

Proposition

Let P be a stratified normal logic program. The locally stratified model of P coincides with the stratified model of P.

Locally Unstratified Logic Program

Extensional Database:

$$man(dilbert) \quad \leftarrow$$

Intensional Database:

$$\begin{aligned} single(dilbert) &\quad \leftarrow \quad man(dilbert), not\ husband(dilbert) \\ husband(dilbert) &\quad \leftarrow \quad man(dilbert), not\ single(dilbert) \end{aligned}$$

The logic program is locally unstratified: