Logic Programming and Deductive Databases

Chapter 10: The Minimal Model

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Summer 2023

http://www.informatik.uni-halle.de/~brass/lp23/

Objectives

After completing this chapter, you should be able to:

- explain why for a set of definite Herbrand clauses, the minimal model is chosen as the semantics.

  And not simply the set of all models.

- determine the minimal Herbrand model of a given program.
- explain and apply the immediate consequence operator T_P.
- explain the significance of the least fixed point of T_P.
- define "supported model" and check a given Herbrand interpretation whether it is a supported model of a logic program.

Contents

Logic Programs (1)

Definition:

- A Logic Program P is a set of definite Horn clauses, i.e. formulas of the form

$$A \leftarrow B_1 \wedge \cdots \wedge B_n.$$

where $A, B_1, \ldots, B_n$ are atomic formulas and $n \geq 0$.

For Prolog execution, the sequence of the rules is important.

Then a Pure Prolog Program is a list of definite Horn clauses.

- Such formulas are called rules. A is called the head of the rule, and $B_1 \wedge \cdots \wedge B_n$ the body of the rule.

Logic Programs (2)

Note:

- In this chapter, we assume that the signature is one-sorted.
- This corresponds to Prolog being untyped, and makes the formalism simpler.
- Often, a signature is not explicitly given (Prolog needs no declarations).
- However, given a logic program P, one can always construct a signature Σ of the symbols that appear in P.

Logic Programs (3)

Definition:

- A fact is a rule with empty body and without variables. The empty body is understood as "true".

  A conjunction is true iff all its conjuncts are true. If there is none, this is trivially satisfied.

- A fact is written as "$A \leftarrow .$" or as "$A.$".
- Sometimes facts are identified with the positive ground literal in the head.
- One also sometimes says "fact" when one really means "positive ground literal" (fact is shorter).

Logic Programs (4)

Note:

- The definitions become simpler when facts are seen as special cases of a rule.
- Of course, in deductive databases one separates
  - predicates that are defined only by facts (EDB predicates: classical relations).

    EDB: Extensional Data Base.
  - predicates that are defined by proper rules (IDB predicates: views).

    IDB: Intensional Data Base.
- In deductive databases, often no function symbols are permitted.

Herbrand Interpretations (1)

- This is only a repetition. See Chapter 3.
- It is difficult to consider arbitrary interpretations.
- Herbrand interpretations have a
  - fixed domain: Set of all ground terms.
  - fixed interpretation of constants as themselves.
  - fixed interpretation of function symbols as term constructors ("free interpretation").
- Thus, only the interpretation of the predicates can be chosen in an Herbrand interpretation.

Herbrand Interpretations (2)

Definition:

- The Herbrand universe $\mathcal{U}_\Sigma$ for a signature Σ is the set of all ground terms that can be constructed with the constants and function symbols in Σ.

  If the signature should contain no constant, one adds one constant "a" (so that the Herbrand universe is not empty).

- For a logic program P, the Herbrand universe $\mathcal{U}_P$ is the set of ground terms that can be built with the constants and function symbols that appear in P.

  I.e. if a signature is not explicitly given, one assumes that the signature contains only the constants and function symbols (and predicates) that appear in P. One must again add a constant if $\mathcal{U}_P$ would otherwise be empty.

Herbrand Interpretations (3)

Definition:

- The Herbrand base $\mathcal{B}_\Sigma$ is the set of all positive ground literals that can be built over Σ.

  Again, one must ensure that the set is not empty be adding a constant if Σ does not contain any constant.

- I.e. the Herbrand base is the set of all formulas of the form $p(t_1, \ldots, t_n)$, where p is a predicate of arity n in Σ, and $t_1, \ldots, t_n \in \mathcal{U}_\Sigma$.

- Again, if instead of a signature Σ, a logic program P is given, one constructs the signature of the symbols that appear in P.

Herbrand Interpretations (4)

- A Herbrand interpretation $\mathcal{I}$ can be identified with the set of all positive ground literals $p(t_1, \ldots, t_n)$ that are true in $\mathcal{I}$, i.e. with $H := \{A \in \mathcal{B}_\Sigma \mid \mathcal{I} \models A\}$.
- Conversely, $H \subseteq \mathcal{B}_\Sigma$ denotes the Herbrand interpretation with

$$\mathcal{I}[p] := \{(t_1, \ldots, t_n) \in \mathcal{U}_\Sigma^n \mid p(t_1, \ldots, t_n) \in H\}.$$

  Otherwise, $\mathcal{I}$ is fixed, because it is a Herbrand interpretation: For the single sort s, $\mathcal{I}[s] := \mathcal{U}_\Sigma$, for constants c, $\mathcal{I}[c] := c$, and for function symbols f of arity n: $\mathcal{I}[f](t_1, \ldots, t_n) := f(t_1, \ldots, t_n)$.

- Thus, in the following, Herbrand interpretations are subsets of $\mathcal{B}_\Sigma$.

Herbrand Interpretations (5)

Definition:

- A Herbrand model of a logic program P is a Herbrand interpretation $\mathcal{I}$ that is a model of P.

Exercise:

- Name two different Herbrand models of P:

$$p(a).$$
$$p(b).$$
$$q(a,b).$$
$$r(X) \leftarrow p(X) \wedge q(X,Y).$$

- Please name also a Herbrand interpretation that is not a Herbrand model of P.

Contents

Minimal Herbrand Model (1)

- A model of a logic program can be "too large" (it can contain unnecessary ground literals).
- The rules enforce only that if the body is true, also the head must be true.
- If the body is false, the rule is automatically satisfied. Nothing is required for the truth of the head.

  That is important because there can be several rules with the same head. If the body of this rule is false, the body of another rule with the same head can be true. Thus, one cannot require that if the body is false, the head must also be false.

Minimal Herbrand Model (2)

- E.g. the entire Herbrand base (the interpretation that makes everything true) is a model of every logic program.
- Of course, one wants a model that contains only those ground literals that must be true because of the rules.
- That is the minimal Herbrand model. It is the declarative semantics of a logic program.

  At least in the area of deductive databases. As we will see, Prolog's SLD-Resolution corresponds more the to set of supported models.

Minimal Herbrand Model (3)

Definition:

- A Herbrand interpretation $\mathcal{I}$ is called Minimal (Herbrand) Model of a logic program P iff
  - $\mathcal{I}$ is model of P ($\mathcal{I} \models P$), and
  - there is no smaller model of P, i.e. no Herbrand interpretation $\mathcal{I}'$ with $\mathcal{I}' \models P$ and $\mathcal{I}' \subset \mathcal{I}$ ($\mathcal{I}' \neq \mathcal{I}$).

Theorem:

- Every logic program has a unique minimal model.

  It is the intersection of all Herbrand models.

Minimal Herbrand Model (4)

Relation to Databases:

- As explained above, a relational database state is an interpretation with finite extensions of the relation symbols and no function symbols.

  For simplicity, we ignore datatype operations.

- If the logic program contains no function symbols, the minimal model is a relational DB state.
- If a predicate is defined only by facts, it is interpreted in the minimal model as exactly these facts.
- Rules then define views (derived predicates).

Minimal Herbrand Model (5)

Theorem:

- Let $\mathcal{I}$ be the minimal Herbrand model of a logic program P.
- For every positive ground literal $A \in \mathcal{B}_\Sigma$:
  If $\mathcal{I} \models A$, then $P \models A$.

  Of course, the same holds for a conjunction of positive ground literals.
- For every positive ground literal $A \in \mathcal{B}_\Sigma$:
  If $\mathcal{I} \not\models A$, then $P \not\models A$.

  This is trivial.

Minimal Herbrand Model (6)

Note:

- The above theorem explains the importance of the minimal model for query evaluation: It is a prototypical model and instead of logical consequence we can talk about truth in this model.
- It does not hold for formulas that contain variables.

  E.g. $P = \{p(a), p(b)\}$. If a and b are the only constants in Σ (and there are no function symbols), $\forall X\ p(X)$ is true in the minimal model, but it is not implied.
- However, in deductive databases, one normally ensures that all variables in the query must be bound.

Minimal Herbrand Model (7)

Exercise:

- What is the minimal model of this logic program?

$$\texttt{mother}(\texttt{alan}, \texttt{bianca}).$$
$$\texttt{father}(\texttt{bianca}, \texttt{chris}).$$
$$\texttt{parent}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{mother}(\texttt{X}, \texttt{Y}).$$
$$\texttt{parent}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{father}(\texttt{X}, \texttt{Y}).$$
$$\texttt{ancestor}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{parent}(\texttt{X}, \texttt{Y}).$$
$$\texttt{ancestor}(\texttt{X}, \texttt{Z}) \leftarrow \texttt{parent}(\texttt{X}, \texttt{Y}) \wedge \texttt{ancestor}(\texttt{Y}, \texttt{Z}).$$

- Guess a model $\mathcal{I}$ and explain for each $A \in \mathcal{I}$ that there cannot be a model without A.

Minimal Herbrand Model (8)

Example:

- Consider the following logic program:

```
a_list([]).
a_list([a|X]) :- a_list(X).
```

- This program has an infinite minimal model:

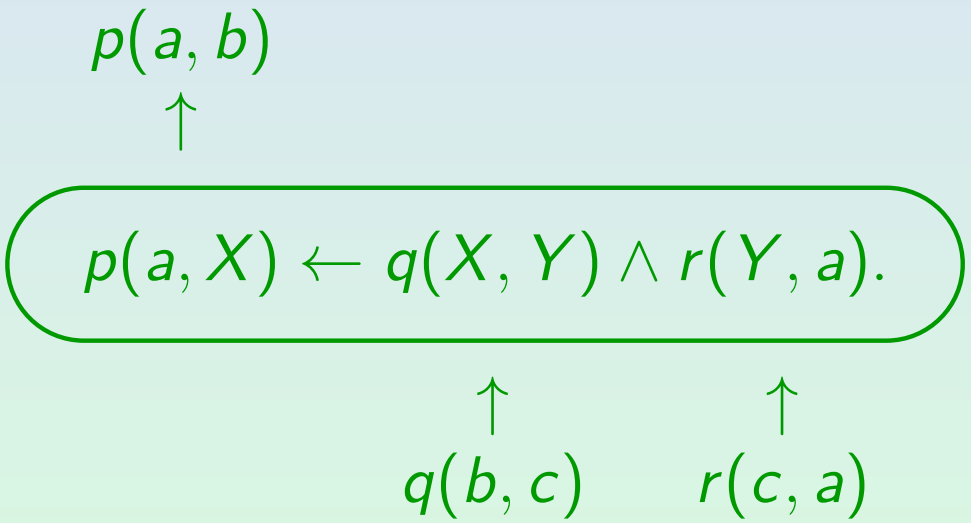
  $\mathcal{I} = \{$`a_list([])`, `a_list([a])`, `a_list([a,a])`, $\ldots\}$.

- This explains e.g. why Prolog answers
  - `a_list([a,a])` with "yes",
  - `a_list([a,b])` with "no".

Contents

Computing the Minimal Model

- One can compute the minimal model by iteratively inserting known facts for the body literals to compute a new literal:



- This rule application is formalized by the immediate consequence operator T_P.

Substitutions (1)

- This is a repetition, see Chapter 3.

  Here the definition is slightly simpler, because no sorts are considered.

- A substitution is a mapping $\theta\colon VARS \to TE_\Sigma$, such that the set $\{V \in VARS \mid \theta(V) \neq V\}$ is finite.

  The restriction ensures that a substitution can be finitely represented. It is not a real restriction because formulas anyway contain only finitely many variables. Note that Σ is here a one-sorted signature, therefore no variable declaration is needed, and TE_Σ contains Terms with arbitrary variables from $VARS$.

- A substitution is usually written down as a set of variable/term-pairs in the form $\{X/a, Y/Z\}$.

  This means the substitution θ with $\theta(X) = a$, $\theta(Y) = Z$, and $\theta(V) = V$ for all other variables V.

Substitutions (2)

- The domain of a substitution can be extended from the set of variables successively to terms, literals, and rules (or arbitrary formulas).
- This is done by replacing the variables inside the term, literal, rule as specified in the substitution and leaving the rest unchanged.
- E.g. the substitution $\theta = \{X/a, Y/Z\}$ applied to the literal $p(X, Y, V, b)$ gives the literal $p(a, Z, V, b)$.
- The postfix notation is often used for applying a substitution, e.g. $A\theta$ means $\theta(A)$.

Substitutions (3)

- Note that a substitution is applied only once, not iteratively. E.g. $\theta = \{X/Y,\ Y/Z\}$ maps $p(X)$ to $p(Y)$, and not to $p(Z)$.
- A substitution θ is a ground substitution for a rule F iff it replaces all variables that occur in F by ground terms.
- Thus, the result of applying a ground substitution to a rule F is a ground rule.

  I.e. a ground substitution replaces all variables by concrete values. For Herbrand interpretations, ground substitutions and variable assignments are basically the same.

Ground Instances (1)

Definition:

- A rule F_1 is an instance of a rule F_2 iff there is a substitution θ with $F_1 = \theta(F_2)$.
- A ground instance is an instance that is variable-free (the result of applying a ground substitution).
- We write $ground(P)$ for the set of all ground instances of rules in P.

Ground Instances (2)

Example:

- E.g. $\texttt{parent}(\texttt{alan}, \texttt{bianca}) \leftarrow \texttt{mother}(\texttt{alan}, \texttt{bianca})$ is a ground instance of $\texttt{parent}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{mother}(\texttt{X}, \texttt{Y})$.
- The ground substitution is $\theta = \{\texttt{X}/\texttt{alan},\ \texttt{Y}/\texttt{bianca}\}$.
- E.g. $\texttt{parent}(\texttt{alan}, \texttt{chris}) \leftarrow \texttt{mother}(\texttt{alan}, \texttt{chris})$ is another ground instance of the same rule.
- E.g. $\texttt{parent}(\texttt{chris}, \texttt{bianca}) \leftarrow \texttt{mother}(\texttt{bianca}, \texttt{doris})$ is not a ground instance of the above rule.

  One must of course replace all occurrences of the same variable in a rule by the same value (when computing a single ground instance of a single rule).

Ground Instances (3)

Exercise:

- Let the following rule be given:

$$p(a, X) \leftarrow q(X, Y) \wedge r(Y, a).$$

- Which of the following rules are ground instances of the given rule?
  - ☐ $p(a, a) \leftarrow q(a, a) \wedge r(a, a).$
  - ☐ $p(a, b) \leftarrow q(a, b) \wedge r(b, a).$
  - ☐ $p(a, b) \leftarrow q(b, c) \wedge r(c, a).$
  - ☐ $p(b, a) \leftarrow q(a, a) \wedge r(a, a).$
  - ☐ $p(a, b) \leftarrow q(b, Y) \wedge r(Y, a).$

T_P-Operator (1)

- Let a logic program P be given.
- The immediate consequence operator T_P maps Herbrand interpretations to Herbrand interpretations:

$$T_P(\mathcal{I}) := \{F \in \mathcal{B}_\Sigma \mid \text{There is a rule } A \leftarrow B_1 \wedge \cdots \wedge B_n \text{ in } P \text{ and a ground substitution } \theta, \text{ such that } \bullet\ B_i\theta \in \mathcal{I} \text{ for } i = 1, \ldots, n, \text{ and } \bullet\ F = A\theta\}.$$

- Note that the case $n = 0$ is possible, then the condition about the body literals is trivially satisfied.

T_P-Operator (2)

- The input interpretation $\mathcal{I}$ consists of facts that are already known (or assumed) to be true.
- The result $T_P(\mathcal{I})$ of the T_P-operator consists of those facts that are derivable in a single step from the given facts and the rules in the program.
- I.e. for each ground instance $A \leftarrow B_1 \wedge \cdots \wedge B_n$ of a rule in P, if the precondition $B_1 \wedge \cdots \wedge B_n$ is true in $\mathcal{I}$ (i.e. $\{B_1, \ldots, B_n\} \subseteq \mathcal{I}$), then $A \in T_P(\mathcal{I})$.

T_P-Operator (3)

Exercise:

- Let the following logic program P be given:

$$p(a, b).$$
$$p(c, c).$$
$$q(X, Y) \leftarrow p(X, Y).$$
$$q(Y, X) \leftarrow p(X, Y).$$

- Let $\mathcal{I}_0 := \emptyset$.
- Please compute $\mathcal{I}_1 := T_P(\mathcal{I}_0)$, $\mathcal{I}_2 := T_P(\mathcal{I}_1)$, and $\mathcal{I}_3 := T_P(\mathcal{I}_2)$.

T_P-Operator (4)

Theorem:

- Let P be any logic program and let
  - $\mathcal{I}_0 := \emptyset$,
  - $\mathcal{I}_{i+1} := T_P(\mathcal{I}_i)$ for $i = 0, 1, \ldots$.
- If there is $n \in \mathbb{N}_0$ with $\mathcal{I}_{n+1} = \mathcal{I}_n$ then $\mathcal{I}_n$ is the minimal Herbrand model of P.
- If $ground(P)$ is finite, there is always such an n.

T_P-Operator (5)

Exercise:

- Please compute the minimal model of the following logic program P by iteratively applying the T_P-operator:

$$\texttt{mother}(\texttt{alan}, \texttt{bianca}).$$
$$\texttt{father}(\texttt{bianca}, \texttt{chris}).$$
$$\texttt{parent}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{mother}(\texttt{X}, \texttt{Y}).$$
$$\texttt{parent}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{father}(\texttt{X}, \texttt{Y}).$$
$$\texttt{ancestor}(\texttt{X}, \texttt{Y}) \leftarrow \texttt{parent}(\texttt{X}, \texttt{Y}).$$
$$\texttt{ancestor}(\texttt{X}, \texttt{Z}) \leftarrow \texttt{parent}(\texttt{X}, \texttt{Y}) \wedge \texttt{ancestor}(\texttt{Y}, \texttt{Z}).$$

- Does $\mathcal{I} \subseteq T_P(\mathcal{I})$ hold for arbitrary $\mathcal{I}$?

Contents

A Little Bit of Lattice Theory (1)

- Let a set $\mathcal{M}$ and a relation $\preceq \subseteq \mathcal{M} \times \mathcal{M}$ be given.
- $\preceq$ is a partial order iff for all $\mathcal{I}, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3 \in \mathcal{M}$ the following holds:
  - $\mathcal{I} \preceq \mathcal{I}$.

    I.e. $\preceq$ is reflexive.
  - $\mathcal{I}_1 \preceq \mathcal{I}_2$ and $\mathcal{I}_2 \preceq \mathcal{I}_3$ implies $\mathcal{I}_1 \preceq \mathcal{I}_3$.

    I.e. $\preceq$ is transitive.
  - $\mathcal{I}_1 \preceq \mathcal{I}_2$ and $\mathcal{I}_2 \preceq \mathcal{I}_1$ implies $\mathcal{I}_1 = \mathcal{I}_2$.

    I.e. $\preceq$ is antisymmetric.
- $(\mathcal{M}, \preceq)$ is then called a partially ordered set.

A Little Bit of Lattice Theory (2)

- Let $(\mathcal{M}, \preceq)$ be a partially ordered set.
- An element $\mathcal{I} \in \mathcal{M}$ is an upper bound of a set $\mathcal{N} \subseteq \mathcal{M}$ iff $\mathcal{J} \preceq \mathcal{I}$ for all $\mathcal{J} \in \mathcal{N}$.
- An element $\mathcal{I} \in \mathcal{M}$ is called least upper bound of a set $\mathcal{N} \subseteq \mathcal{M}$ iff
  - it is an upper bound, i.e. $\mathcal{J} \preceq \mathcal{I}$ for all $\mathcal{J} \in \mathcal{N}$,
  - $\mathcal{I} \preceq \mathcal{I}'$ for all $\mathcal{I}' \in \mathcal{M}$ that are an upper bound of $\mathcal{N}$.
- In the same way, one defines lower bound and greatest lower bound.

A Little Bit of Lattice Theory (3)

Lemma:

- If the least upper bound exists, it is unique.

Definition (Complete Lattice):

- A partially ordered set $(\mathcal{M}, \preceq)$ is a complete lattice if and only if
  - for every $\mathcal{N} \subseteq \mathcal{M}$ there is a least upper bound and a greatest lower bound.
- Then one writes $lub(\mathcal{N})$ for the least upper bound and $glb(\mathcal{N})$ for the greatest lower bound.

A Little Bit of Lattice Theory (4)

Definition (Top and Bottom Elements):

- A complete lattice $(\mathcal{M}, \preceq)$ always contains
  - a top element $\top := lub(\mathcal{M})$, and
  - a bottom element $\bot := glb(\mathcal{M})$.

Example:

- The set of all Herbrand interpretations (over a fixed signature) together with $\subseteq$ is a complete lattice:

$$lub(\mathcal{N}) = \bigcup_{\mathcal{I} \in \mathcal{N}} \mathcal{I}, \quad glb(\mathcal{N}) = \bigcap_{\mathcal{I} \in \mathcal{N}} \mathcal{I}, \quad \bot = \emptyset, \quad \top = \mathcal{B}_\Sigma.$$

A Little Bit of Lattice Theory (5)

Definition (Properties of Mappings):

- Let $T : \mathcal{M} \to \mathcal{M}$.
- T is monotonic iff $T(\mathcal{I}_1) \preceq T(\mathcal{I}_2)$ for all $\mathcal{I}_1 \preceq \mathcal{I}_2$.
- T is continuous iff $T\Big(lub(\mathcal{N})\Big) = lub\Big(T(\mathcal{N})\Big)$ for all $\mathcal{N} \subseteq \mathcal{M}$ such that every finite subset of $\mathcal{N}$ has an upper bound in $\mathcal{N}$. Here $T(\mathcal{N}) := \{T(\mathcal{I}) \mid \mathcal{I} \in \mathcal{N}\}$.

Lemma:

- If T is continuous, it is also monotonic.

A Little Bit of Lattice Theory (6)

Definition (Fixpoints):

- $\mathcal{I} \in \mathcal{M}$ is a fixpoint of T iff $T(\mathcal{I}) = \mathcal{I}$.
- $\mathcal{I} \in \mathcal{M}$ is the least fixpoint of T iff $T(\mathcal{I}) = \mathcal{I}$ and $\mathcal{I} \preceq \mathcal{J}$ for all $\mathcal{J} \in \mathcal{M}$ with $T(\mathcal{J}) = \mathcal{J}$.

Theorem:

- A monotonic mapping T in a complete lattice has always a least fixpoint, namely

$$glb\Big(\{\mathcal{I} \in \mathcal{M} \mid T(\mathcal{I}) \preceq \mathcal{I}\}\Big).$$

Let $lfp(T)$ be the least fixpoint of T.

A Little Bit of Lattice Theory (7)

Lemma:

- The immediate consequence operator T_P is monotonic and even continuous.

Lemma:

- $\mathcal{I}$ is a model of P iff $T_P(\mathcal{I}) \subseteq \mathcal{I}$.

Theorem:

- The least fixpoint of T_P is the minimal model of P.

A Little Bit of Lattice Theory (8)

Definition (Iteration of a Mapping):

- $T \uparrow 0 := \bot$.
- $T \uparrow (n+1) := T(T \uparrow n)$.
- $T \uparrow \omega := lub\Big(\{T \uparrow n \mid n \subseteq \mathbb{N}_0\}\Big)$.

Note:

- If there is $n \in \mathbb{N}_0$ with $T \uparrow (n+1) = T \uparrow n$, then $T \uparrow m = T \uparrow n$ for all $m \geq n$ and thus $T \uparrow \omega = T \uparrow n$.
- $T \uparrow \gamma$ can be defined for arbitrary ordinal numbers γ.

  $T \uparrow \gamma := T\Big(T \uparrow (\gamma - 1)\Big)$ if γ is successor of $\gamma - 1$, and
  $T \uparrow \gamma := lub\Big(\{T \uparrow \beta \mid \beta < \gamma\}\Big)$ otherwise (γ is a limit ordinal).

A Little Bit of Lattice Theory (9)

Lemma:

- If T is monotonic, $T \uparrow i \preceq T \uparrow (i+1)$ for all $i \in \mathbb{N}_0$.

Theorem:

- If T is continuous, it holds that $lfp(T) = T \uparrow \omega$.

  This implies the theorem on page 33 (minimal model computation).

  If T is only monotonic, there is an ordinal number γ with $lfp(T) = T \uparrow \gamma$.

Corollary:

- Even when the iteration of the T_P-operator does not terminate, every fact that is true in the minimal model is generated after finitely many iterations.

Contents

Supported Models (1)

Supported Model:

- A Herbrand model $\mathcal{I}$ of P is called supported model of P iff $T_P(\mathcal{I}) = \mathcal{I}$ (i.e. $\mathcal{I}$ is a fixpoint of T_P).

Note:

- Thus, for every fact A that is true in $\mathcal{I}$ there is a reason in form of a ground instance

$$A \leftarrow B_1 \wedge \cdots \wedge B_n$$

of a rule in P that permits to derive A (because $\mathcal{I} \models B_i$ for $i = 1, \ldots, n$).

Supported Models (2)

Corollary:

- The minimal model is a supported model.

Note:

- The converse is not true: Consider e.g. the program

$$P := \{p \leftarrow p\}.$$

The interpretation $\mathcal{I} := \{p\}$ is a supported model of P, but it is not minimal.

Practical example: $\texttt{married\_with(X, Y)} \leftarrow \texttt{married\_with(Y, X)}.$

- However, non-recursive programs (see below) have only one supported model, namely the minimal model.