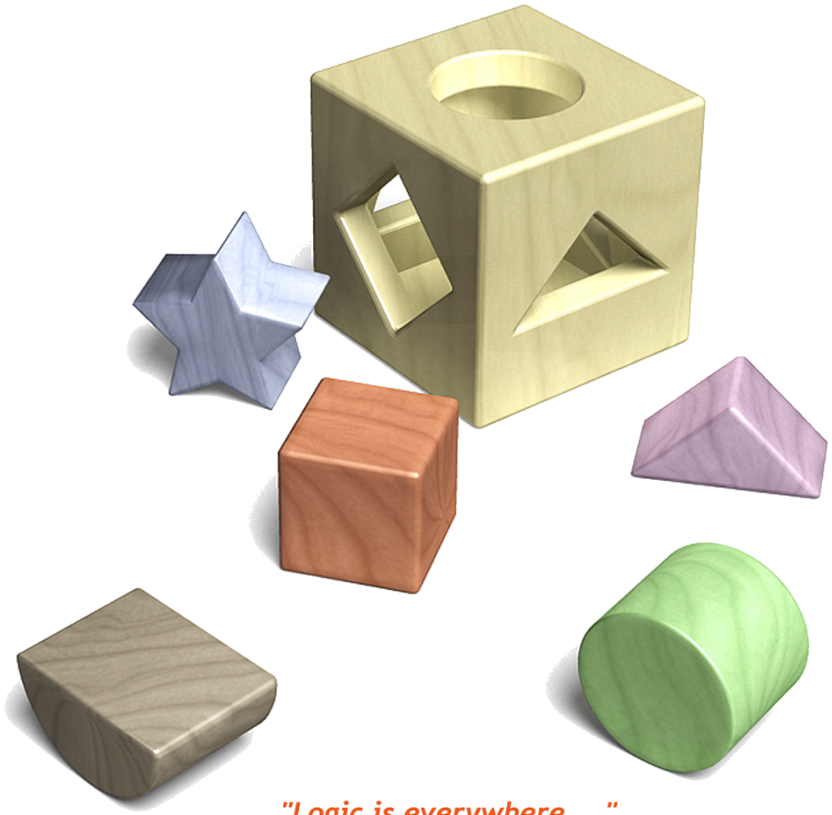
Foundations of Logic Programming

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- **Definite Logic Programs**
- **Model Theoretic Semantics**
- **Fixed Point Semantics**



"Logic is everywhere ..."

Literature

- Hölldobler: *Logik und Logikprogrammierung, vol 1: Grundlagen* Synchron Verlag, Heidelberg: 2009

  In this chapter, the numbering of theorems refers to this book

- Bader etal.: *Logik und Logikprogrammierung, vol 2: Aufgaben und Lösungen* Synchron Verlag, Heidelberg: 2011
- Lloyd: *Foundations of Logic Programming*. Springer Verlag, Berlin: 1984
- Apt, van Emden: *Contributions to the Theory of Logic Programming*. JACM 29, 841-862: 1982

Logic Programming

- Restriction of the resolution rule
  - efficient implementation
- Strong connection between program clauses and procedures
  - programming
  - declarative semantics
- Computational adequateness
  - every computable function can be expressed by a definite logic program

Definite Program Clauses

- **Recall clauses**

$$[A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m]$$

  where A_i, $0 \leq i \leq n$, and B_j, $0 \leq j \leq m$, are atoms

  - **A_i positive**
  - **$\neg B_j$ negative literals**

- **Definition**

  - **A definite program clause is a clause which contains precisely one positive literal**
  - **A unit clause or fact is a definite program clause which does not contain negative literals**

Notation

- We find

$$\begin{aligned}[A, \neg B_1, \ldots, \neg B_m] &= (A \vee [\neg B_1, \ldots, \neg B_m]) \\ &\equiv (A \vee \neg\langle B_1, \ldots, B_m\rangle) \\ &\equiv (A \leftarrow \langle B_1, \ldots, B_m\rangle)\end{aligned}$$

- We denote definite program clauses by

$$A \leftarrow B_1 \wedge \ldots \wedge B_m$$

  and facts by

$$A$$

  assuming that $\wedge$ binds stronger than $\leftarrow$

- In the programming language Prolog program clauses are denoted as

```
A :- B1, ..., Bm.
A.
```


Definite Programs

- **Definition** Let $A \leftarrow B_1 \wedge \ldots \wedge B_m$ be a definite program clause
  - A is called **head**
  - $B_1 \wedge \ldots \wedge B_m$ is called **body** of the program clause
- **Definition** A **definite program** is a first-order logic sentence $\forall \langle C_1, \ldots, C_\ell \rangle$ in clause form, where each C_j, $1 \leq j \leq \ell$, is a definite program clause
- **Notation**
  - Quantifiers and angular brackets are omitted
  - Program clauses are written one above the other
  - In the sequel $\mathcal{P}$ denotes a definite program

Example

$\mathcal{P}$

$$append([\,], L, L)$$
$$append([K|R], L, [K|L_1]) \leftarrow append(R, L, L_1)$$

$$shuffle(L, [\,], L)$$
$$shuffle(L, [K|R], S) \leftarrow append(L_1, L_2, L) \wedge shuffle(L_2, R, S_1) \wedge append(L_1, [K|S_1], S)$$

Calls

$$\leftarrow append([1, 2], [3, 4, 5], Z)$$

$$\leftarrow append(X, [3, 4, 5], [1, 2, 3, 4, 5])$$

$$\leftarrow append(X, Y, [1, 2, 3, 4, 5])$$

$$\leftarrow append(X, Y, Z)$$

$$\leftarrow shuffle([a, b, c], [1, 2, 3], X)$$

Definite Goals

- **Definition** A definite goal is a clause containing only negative literals
  Each of these negative literals is called subgoal
- **Notation**
  - Because

$$\begin{aligned}[\neg B_1, \ldots, \neg B_m] &\equiv ([\,] \vee [\neg B_1, \ldots, \neg B_m]) \\ &\equiv ([\,] \vee \neg\langle B_1, \ldots, B_m\rangle) \\ &\equiv ([\,] \leftarrow \langle B_1, \ldots, B_m\rangle)\end{aligned}$$

    we write goals in the form

$$\leftarrow B_1 \wedge \ldots \wedge B_m$$

  - In the sequel G denotes a definite goal
- The empty clause [ ] is a definite goal
- In Prolog definite goals are denoted by

$$?- B_1, \ldots, B_m.$$

Goals as Queries

- Let $\mathcal{P} = \forall \langle C_1, \ldots, C_\ell \rangle$ be a definite program
- We find

$$
\begin{array}{lll}
\mathcal{P} \models \exists \langle B_1, \ldots, B_m \rangle & & \\
\textbf{iff} \quad \mathcal{P} \rightarrow \exists \langle B_1, \ldots, B_m \rangle & & \text{valid (Theorem 3.17)} \\
\textbf{iff} \quad \neg(\mathcal{P} \rightarrow \exists \langle B_1, \ldots, B_m \rangle) & & \text{unsatisfiable (Theorem 3.14)} \\
\textbf{iff} \quad \neg(\neg\mathcal{P} \vee \exists \langle B_1, \ldots, B_m \rangle) & & \text{unsatisfiable (Theorem 3.19)} \\
\textbf{iff} \quad \neg\neg\mathcal{P} \wedge \neg\exists \langle B_1, \ldots, B_m \rangle & & \text{unsatisfiable (Theorem 3.19)} \\
\textbf{iff} \quad \mathcal{P} \wedge \neg \exists \langle B_1, \ldots, B_m \rangle & & \text{unsatisfiable (Theorem 3.19)} \\
\textbf{iff} \quad \mathcal{P} \wedge \forall \neg\langle B_1, \ldots, B_m \rangle & & \text{unsatisfiable (Theorem 4.32)} \\
\textbf{iff} \quad \mathcal{P} \wedge \forall (\leftarrow B_1 \wedge \ldots \wedge B_m) & & \text{unsatisfiable (Theorem 3.19)} \\
\textbf{iff} \quad \forall \langle C_1, \ldots, C_\ell, \leftarrow B_1 \wedge \ldots \wedge B_m \rangle & & \text{unsatisfiable (Theorem 4.32)}
\end{array}
$$

- Goals are queries concerning the existence of objects which meet certain conditions

Horn Clauses

- **Definition** **A Horn clause is a definite program clause or a definite goal**
- **We restrict ourselves to Horn clauses in this chapter**

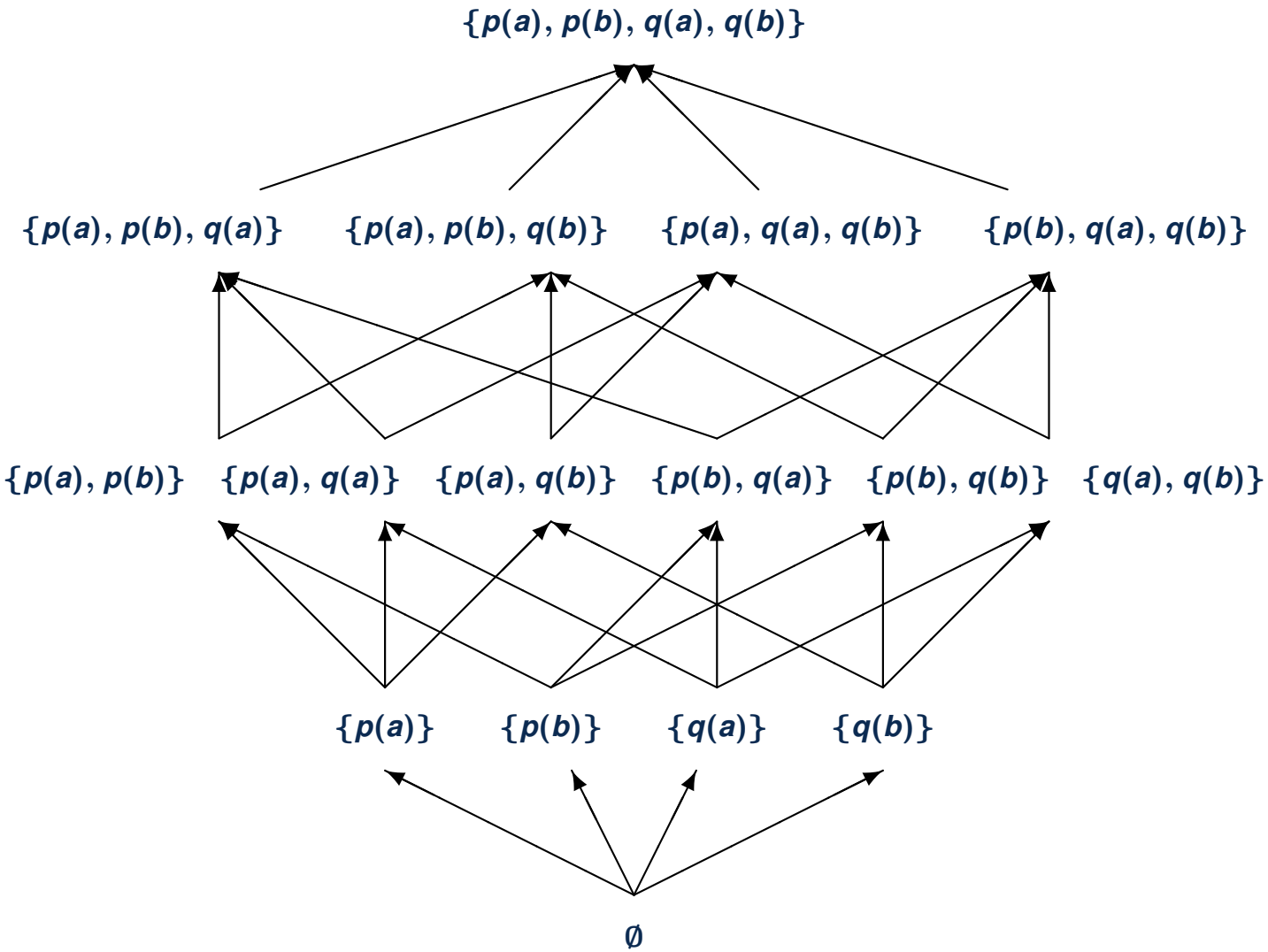
Semantics

- Let $\mathcal{P}$ be a definite program over $\mathcal{R}$, $\mathcal{F}$, and $\mathcal{V}$
  - $\mathcal{F}$ must contain at least one constant symbol
  - $\mathcal{P}$ is in Skolem normal form
  - We can restrict interpretations to Herbrand interpretations (Theorem 4.60)
- The set $\mathcal{T}(\mathcal{F})$ of closed terms is called Herbrand universe
- The set $\mathcal{A}(\mathcal{R}, \mathcal{F})$ of closed atoms is called Herbrand base
  - Each Herbrand interpretation can be represented by some $I \subseteq \mathcal{A}(\mathcal{R}, \mathcal{F})$ and vice versa
  - $2^{\mathcal{A}(\mathcal{R},\mathcal{F})}$ is the set of all Herbrand interpretations
  - $(2^{\mathcal{A}(\mathcal{R},\mathcal{F})}, \subseteq)$ is a complete lattice

Example

- Let $\mathcal{R} = \{p/1,\ q/1\}$ and $\mathcal{F} = \{a/0, b/0\}$
  - $\mathcal{T}(\mathcal{F}) = \{a,\ b\}$
  - $\mathcal{A}(\mathcal{R}, \mathcal{F}) = \{p(a),\ p(b),\ q(a),\ q(b)\}$
  - There are $2^4 = 16$ different Herbrand interpretations
- Let $\mathcal{P}$ be the program consisting of $p(a)$
  $q(X) \leftarrow p(X)$

The Complete Lattice of Herbrand Interpretations



Example Continued

- Let $\mathcal{P}$ be the program consisting of $p(a)$ and $q(X) \leftarrow p(X)$
  - $I_1 = \mathcal{A}(\mathcal{R}, \mathcal{F}) = \{p(a),\ p(b),\ q(a),\ q(b)\}$ is a model for $\mathcal{P}$
  - $I_2 = \{p(a),\ q(a),\ q(b)\}$ is a model for $\mathcal{P}$
- **Observation**
  - $I_1 \cap I_2$ is a model for $\mathcal{P}$
  - This holds generally:
    The intersection of two models for $\mathcal{P}$ is a model for $\mathcal{P}$

Least Models

- **Proposition** Let $\mathcal{M}$ be a non-empty set of Herbrand models for $\mathcal{P}$
  Then, $\bigcap_{I \in \mathcal{M}} I$ is also a Herbrand model for $\mathcal{P}$
- **Proof** Let $\mathcal{M}$ be a non-empty set of Herbrand models for $\mathcal{P}$
  - $J := \bigcap_{I \in \mathcal{M}} I$ is a Herbrand interpretation for $\mathcal{P}$
  - **To show** $J \models \mathcal{P}$
  - Suppose $J \not\models \mathcal{P}$
  - Then we find a ground instance $A \leftarrow B_1 \wedge \ldots \wedge B_m$ of a clause in $\mathcal{P}$ with $\{B_1, \ldots, B_m\} \subseteq J$ and $A \notin J$
  - Then we find $I \in \mathcal{M}$ with $\{B_1, \ldots, B_m\} \subseteq I$ and $A \notin I$
  - Hence $I \not\models \mathcal{P}$ **contradiction**
  - Thus $J \models \mathcal{P}$
- **Notation** $M_{\mathcal{P}}$ denotes the least Herbrand model for $\mathcal{P}$

Model Theoretic Semantics

- **Theorem** $M_{\mathcal{P}} = \{A \mid A \in \mathcal{A}(\mathcal{R}, \mathcal{F}) \text{ and } \mathcal{P} \models A\}$
- **Proof**

$\mathcal{P} \models A$	iff	$\models \mathcal{P} \rightarrow A$	(Theorem 3.17)
	iff	$\neg(\mathcal{P} \rightarrow A)$ is unsatisfiable	(Theorem 3.14)
	iff	$\neg(\neg\mathcal{P} \vee A)$ is unsatisfiable	(Theorem 3.19)
	iff	$\neg\neg\mathcal{P} \wedge \neg A$ is unsatisfiable	(Theorem 3.19)
	iff	$\mathcal{P} \wedge \neg A$ is unsatisfiable	(Theorem 3.19)
	iff	$\mathcal{P} \wedge \neg A$ has no Herbrand models	(Theorem 4.60)
	iff	$\neg A$ is false under all Herbrand models of $\mathcal{P}$	
	iff	A is true under all Herbrand models of $\mathcal{P}$	
	iff	$A \in M_{\mathcal{P}}$	

A Semantic Operator

- **Remember**
  - $(2^{\mathcal{A}(\mathcal{R},\mathcal{F})}, \subseteq)$ is a complete lattice
  - Each montone mapping T over a complete lattice has a least fixed point
    - It is denoted by $lfp(T)$
- **Definition** Let I be a Herbrand interpretation
  The mapping $T_{\mathcal{P}} : 2^{\mathcal{A}(\mathcal{R},\mathcal{F})} \rightarrow 2^{\mathcal{A}(\mathcal{R},\mathcal{F})}$ is defined by

$$T_{\mathcal{P}}(I) = \{A \mid A \leftarrow B_1 \wedge \ldots \wedge B_m \text{ is the ground instance of a clause in } \mathcal{P} \text{ and } \{B_1, \ldots, B_n\} \subseteq I\}$$

- **Example** Let $\mathcal{P}$ be $p(a)$, $r(f(f(X))) \leftarrow p(X)$, $r(f(Y)) \leftarrow r(Y)$

Then

$$\begin{aligned} T_{\mathcal{P}}(\emptyset) &= \{p(a)\} \\ T_{\mathcal{P}}(\{p(a)\}) &= \{p(a),\ r(f(f(a)))\} \\ T_{\mathcal{P}}(\{p(a),\ r(f(f(a)))\}) &= \{p(a),\ r(f(f(a))),\ r(f(f(f(a))))\} \end{aligned}$$

Fixed Point Semantics

- **Proposition** $T_{\mathcal{P}}$ is monotone and continuous
- **Theorem** $M_{\mathcal{P}} = lfp(T_{\mathcal{P}})$
- The least fixed point of $T_{\mathcal{P}}$ can be computed as follows: Let

$$\begin{aligned} T_{\mathcal{P}} \uparrow 0 &= \emptyset \\ T_{\mathcal{P}} \uparrow (n+1) &= T_{\mathcal{P}}(T_{\mathcal{P}} \uparrow n) \quad \text{if } n \in \mathbb{N} \end{aligned}$$

Then $lfp(T_{\mathcal{P}}) = lub(\{T_{\mathcal{P}} \uparrow n \mid n \in \mathbb{N}\})$
where *lub* denotes the least upper bound

- **Example** Let $\mathcal{P}$ be $p(a)$
  $r(f(f(X))) \leftarrow p(X)$
  $r(f(Y)) \leftarrow r(Y)$

Then $lfp(T_{\mathcal{P}}) = \{p(a)\} \cup \{r(f^i(a)) \mid i \geq 2\}$

Some Properties

- **Theorem (Undecidability)**
  The question 'is $\mathcal{P} \wedge G$ unsatisfiable' is undecidable
- **Theorem (Universality)**
  Each partially recursive function can be computed by a definite program
- **Theorem (Monotonicity)**

  Let $\mathcal{P}$ and $\mathcal{P}'$ be definite programs and F a sentence
  If $\mathcal{P} \models F$ then $\mathcal{P} \wedge \mathcal{P}' \models F$

Extensions

- We will allow negation in the body of clauses

$$a \leftarrow b \wedge \neg c$$

- It is sometimes convenient to introduce the symbols $\top$ and $\bot$ denoting truth and falsehood, respectively, i.e. $I(\top) = \top$ and $I(\bot) = \bot$ for all interpretations I
- Then, facts can be written in the form

$$A \leftarrow \top$$

- We will allow assumptions of the form

$$A \leftarrow \bot$$

- We will consider three-valued logics mapping formulas to $\{\top, \bot, \mathbf{U}\}$
- We will consider non-monotonic logics