Hannes Strass (based on slides by Michael Thielscher)
Faculty of Computer Science, Institute of Artificial Intelligence, Computational Logic Group

Least Herbrand Models

Lecture 5, 6th Nov 2023 // Foundations of Logic Programming, WS 2023/24

Previously ...

- The semantics of (definite) logic programs is given by a standard first-order model theory.
- SLD resolution is **sound**: For every successful SLD derivation of $P \cup \{Q_0\}$ with *computed* answer substitution θ, we have $P \models Q_0\theta$.
- SLD resolution is **complete**: If θ is a *correct* answer substitution of Q, then
  - for every selection rule
  - there exists a successful SLD derivation of $P \cup \{Q\}$ with cas η
  - such that $Q\eta$ is more general than $Q\theta$.

$$P \vdash_{SLD} Q_0\eta \quad \Longleftrightarrow \quad P \models Q_0\theta$$

η more general than θ

proof theory — model theory

Ground Implication Trees Constitute Herbrand Models

Lemma 4.26

Consider Herbrand interpretation I, atom A, program P.

- $I \models A$ iff $ground(A) \subseteq I$
- $I \models P$ iff for every $A \leftarrow B_1, \ldots, B_n \in ground(P)$,

$$\{B_1, \ldots, B_n\} \subseteq I \text{ implies } A \in I$$

Lemma 4.28

The Herbrand interpretation

$$\mathcal{M}(P) := \{ A \mid A \text{ is the root of some ground implication tree w.r.t. } P \}$$

is a model of P.

Overview

Least Herbrand Models

Computing Least Herbrand Models

History

Turing-Completeness

Least Herbrand Models

Least Herbrand Model (1)

Theorem (Model Intersection Property)

Let P be a definite logic program and $\mathcal{K}$ be a non-empty set of Herbrand models of P. Then $\bigcap \mathcal{K}$ is again a Herbrand model of P.

Proof.

- Employing Lemma 4.26, assume that $A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P)$.
- If $\{B_1, \ldots, B_n\} \subseteq \bigcap \mathcal{K}$, then for each $K \in \mathcal{K}$ we have $\{B_1, \ldots, B_n\} \subseteq K$.
- Thus for each $K \in \mathcal{K}$, since K is a Herbrand model of P, we get $A \in K$.
- Hence $A \in K$ for each $K \in \mathcal{K}$, thus $A \in \bigcap \mathcal{K}$. $\square$

Note: This property does not hold for (sets of) general (non-Horn) clauses.

Corollary

The set $\bigcap \{ I \mid I$ is a Herbrand model of P $\}$ is the least Herbrand model of P.

Least Herbrand Model (2)

Theorem 4.29

$\mathcal{M}(P)$ is the least Herbrand model of P.

Proof.

Let I be a Herbrand model of P and let $A \in \mathcal{M}(P)$.
We prove $A \in I$ by induction on the number i of nodes in the ground implication tree w.r.t. P with root A. It then follows that $\mathcal{M}(P) \subseteq I$.

$i = 1$: A is a leaf implies $A \leftarrow \; \in ground(P)$
implies $I \models A$ (since $I \models P$)
implies $A \in I$

$i \rightsquigarrow i+1$: A has direct descendants $B_1, \ldots, B_n$ (roots of subtrees)
implies $A \leftarrow B_1, \ldots, B_n \in ground(P)$ and $B_1, \ldots, B_n \in I$ (I.H.)
implies $A \leftarrow B_1, \ldots, B_n \in ground(P)$ and $I \models B_1, \ldots, B_n$
implies $I \models A$ (since $I \models P$)
implies $A \in I$ □

Ground Equivalence

Theorem 4.30

For every ground atom A: $P \models A$ if and only if $\mathcal{M}(P) \models A$.

Proof.

"$\Rightarrow$": $P \models A$ and $\mathcal{M}(P) \models P$ implies $\mathcal{M}(P) \models A$ (semantic consequence).

"$\Leftarrow$": Let $A \in \mathcal{M}(P)$. Show for every interpretation I: $I \models P$ implies $I \models A$.

Define $I_H = \{A \mid A \text{ ground atom and } I \models A\}$ the Herbrand interpretation of I.

$$
\begin{array}{ll}
 & I \models P \\
\text{implies} & I \models A \leftarrow B_1, \ldots, B_n \text{ for all } c = A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P) \\
\text{implies} & \text{if } I \models B_1, \ldots, I \models B_n \text{ then } I \models A \text{ for all } c \in \mathit{ground}(P) \\
\text{implies} & \text{if } B_1 \in I_H, \ldots, B_n \in I_H \text{ then } A \in I_H \text{ for all } c \in \mathit{ground}(P) \text{ (Def. } I_H) \\
\text{implies} & I_H \models P \text{ (by Lemma 4.26; thus } I_H \text{ is a Herbrand model of } P) \\
\text{implies} & A \in I_H \text{ (since } A \in \mathcal{M}(P) \text{ and } \mathcal{M}(P) \text{ least Herbrand model of } P) \\
\text{implies} & I \models A \text{ (by Def. } I_H)
\end{array}
$$

□

Computing Least Herbrand Models

Complete Partial Orders

Definition

Let $(\mathcal{A}, \sqsubseteq)$ be a partially ordered set, i.e. $\sqsubseteq \subseteq \mathcal{A} \times \mathcal{A}$. (cf. Lecture 2)

- $a \in \mathcal{A}$ is the **least** element of $X \subseteq \mathcal{A}$ $:\Longleftrightarrow$ $a \in X$ and $a \sqsubseteq x$ for all $x \in X$
- $b \in \mathcal{A}$ is an **upper bound** of $X \subseteq \mathcal{A}$ $:\Longleftrightarrow$ $x \sqsubseteq b$ for all $x \in X$
- $a \in \mathcal{A}$ is the **least upper bound** of $X \subseteq \mathcal{A}$ (Notation: $a = \bigsqcup X$) $:\Longleftrightarrow$ a is the least element of $\{b \in \mathcal{A} \mid b \text{ is an upper bound of } X\}$

Definition

The pair $(\mathcal{A}, \sqsubseteq)$ is a **complete** partial order (**cpo**) $:\Longleftrightarrow$

- $\mathcal{A}$ contains a least element (denoted by $\emptyset$),
- for every ascending chain $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \ldots$ of elements of $\mathcal{A}$, the set $X = \{a_0, a_1, a_2, \ldots\}$ has a least upper bound.

Some Properties of Operators

Definition

Let $(\mathcal{A}, \sqsubseteq)$ be a CPO and $T\colon \mathcal{A} \to \mathcal{A}$ be an operator.

- T is **monotonic** (or **order-preserving**)
  $:\Longleftrightarrow$ for all $I_1, I_2 \in \mathcal{A}$: $I_1 \sqsubseteq I_2$ implies $T(I_1) \sqsubseteq T(I_2)$
- T is **finitary** $:\Longleftrightarrow$ for every infinite ascending chain $I_0 \sqsubseteq I_1 \sqsubseteq \ldots$,
  $$\bigsqcup \{T(I_0), T(I_1), \ldots\} \text{ exists} \quad \text{and} \quad T\left(\bigsqcup \{I_0, I_1, \ldots\}\right) \sqsubseteq \bigsqcup \{T(I_0), T(I_1), \ldots\}$$
- T is **continuous** $:\Longleftrightarrow$ T is monotonic and finitary

Intuitively, a continuous operator preserves least upper bounds:

$$T\left(\bigsqcup \{I_0, I_1, \ldots\}\right) = \bigsqcup \{T(I_0), T(I_1), \ldots\}$$

The other inclusion follows from T being monotone: Since $I_0 \sqsubseteq I_1 \sqsubseteq \ldots$ is a chain and T is monotone, $T(I_0) \sqsubseteq T(I_1) \sqsubseteq \ldots$ is again a chain and $\bigsqcup \{T(I_0), T(I_1), \ldots\}$ exists. Since $I_i \sqsubseteq \bigsqcup \{I_0, I_1, \ldots\}$ for any $i \in \mathbb{N}$ and T is monotone, $T(I_i) \sqsubseteq T(\bigsqcup \{I_0, I_1, \ldots\})$. Thus $T(\bigsqcup \{I_0, I_1, \ldots\})$ is an upper bound of $\{T(I_0), T(I_1), \ldots\}$ and $\bigsqcup \{T(I_0), T(I_1), \ldots\} \sqsubseteq T(\bigsqcup \{I_0, I_1, \ldots\})$.

Iterating Operators

Definition

Let $(\mathcal{A}, \sqsubseteq)$ be a CPO, $T \colon \mathcal{A} \to \mathcal{A}$, and $I \in \mathcal{A}$.

$$T{\uparrow}0\,(I) := I$$
$$T{\uparrow}(n+1)\,(I) := T(T{\uparrow}n(I))$$
$$T{\uparrow}\omega\,(I) := \bigsqcup \{T{\uparrow}n(I) \mid n \in \mathbf{N}\}$$

Similarly, define

$$T{\uparrow}\alpha := T{\uparrow}\alpha\,(\emptyset) \qquad \text{for } \alpha = 0, 1, 2, \ldots, \omega$$

By the definition of a complete partial order:
If the sequence $T{\uparrow}0\,(I), T{\uparrow}1\,(I), T{\uparrow}2\,(I), \ldots$ is increasing, then $T{\uparrow}\omega\,(I)$ exists.

Fixpoints and Pre-Fixpoints

Definition

Let $T \colon \mathcal{A} \to \mathcal{A}$ be an operator and $I \in \mathcal{A}$.

- I is a **pre-fixpoint** of T $\quad :\Longleftrightarrow \quad T(I) \sqsubseteq I$
- I is a **fixpoint** of T $\quad :\Longleftrightarrow \quad T(I) = I$

Theorem 4.22 (Kleene's fixpoint theorem)

If T is a continuous operator on a CPO $(\mathcal{A}, \sqsubseteq)$, then $T{\uparrow}\omega$ exists and is the least fixpoint of T.

Proposition 4.23

Let $(\mathcal{A}, \sqsubseteq)$ be a partially ordered set and $T \colon \mathcal{A} \to \mathcal{A}$ be a monotone operator. If T has a least pre-fixpoint π, then π is also the least fixpoint of T.

Proof Idea: If π is the least element of $\{\rho \in \mathcal{A} \mid T(\rho) \sqsubseteq \rho\}$ then $T(T(\pi)) \sqsubseteq T(\pi)$ since T is monotone, thus $\pi \sqsubseteq T(\pi)$, that is, $T(\pi) = \pi$.

One-Step Consequence Operator

Definition

Consider the cpo $(\mathcal{I}, \subseteq)$ with $\mathcal{I} = \{I \mid I \text{ is a Herbrand interpretation}\}$.
Let P be a program. Define the operator $T_P \colon \mathcal{I} \to \mathcal{I}$ as follows:

$$T_P(I) := \{\, A \mid A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P), \{B_1, \ldots, B_n\} \subseteq I \,\}$$

Lemma 4.33

Let P be a program.

(i) T_P is finitary.

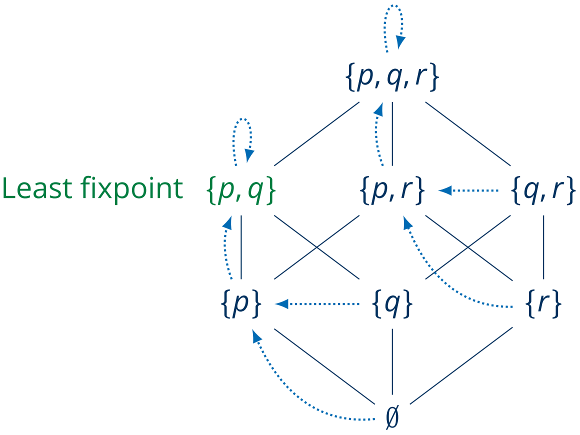
(ii) T_P is monotonic.

Thus T_P is continuous and its least fixpoint is given by $T_P{\uparrow}\omega = T_P{\uparrow}\omega(\emptyset)$.

T_P-Operator: Example (1)

Consider the (propositional) program $P = \{p \leftarrow, \quad q \leftarrow p, \quad r \leftarrow r\}$.

The operator T_P maps as follows: $I \dashrightarrow T_P(I)$



Quiz: T_P-Operator

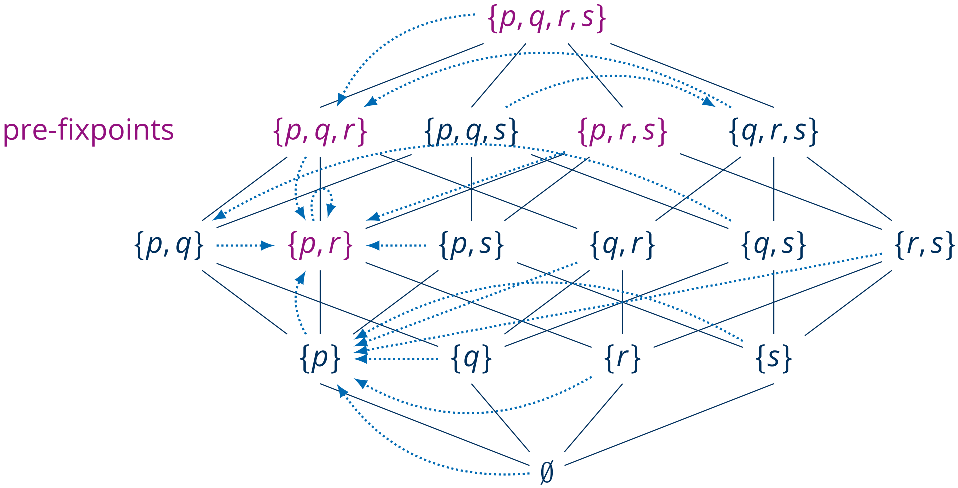
Recall: $T_P(I) := \{ A \mid A \leftarrow B_1, \ldots, B_n \in ground(P), \{B_1, \ldots, B_n\} \subseteq I \}$.

Quiz

Consider the following (definite) logic program: …

T_P-Operator: Example (2)

Consider the logic program $P = \{p \leftarrow, \quad q \leftarrow q, s, \quad r \leftarrow p\}$.



T_P-Characterization

Lemma 4.32

A Herbrand interpretation I is a model of P iff

$$T_P(I) \subseteq I$$

Proof.

$$\begin{array}{ll} & I \models P \\ \text{iff} & \text{for every } A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P)\text{:} \\ & \{B_1, \ldots, B_n\} \subseteq I \text{ implies } A \in I \qquad \text{(by Lemma 4.26)} \\ \text{iff} & \text{for every ground atom } A\text{: } A \in T_P(I) \text{ implies } A \in I \\ \text{iff} & T_P(I) \subseteq I \end{array}$$

□

Characterization Theorem

Theorem 4.34

	$\{ A \mid A \text{ ground atom}, P \models A \}$	
$=$	$\mathcal{M}(P)$	(Theorem 4.30)
$=$	least Herbrand model of P	(Theorem 4.29)
$=$	least pre-fixpoint of T_P	(Lemma 4.32)
$=$	least fixpoint of T_P	(Proposition 4.23)
$=$	$T_P \uparrow \omega$	(Theorem 4.22)

Success Sets

Definition

The **success set** of a program P is the set of all ground atoms A for which there exists a successful SLD derivation of $P \cup \{A\}$.

Theorem 4.37

For a ground atom A, the following are equivalent:

(i) $\mathcal{M}(P) \models A$

(ii) $P \models A$

(iii) Every SLD tree for $P \cup \{A\}$ is successful

(iv) A is in the success set of P

History

Timeline

1965: John Alan Robinson: The resolution

1970: Alain Colmerauer: Q-systems

197

197

197

Groupe de recherche en

Intelligence Artificielle

U.E.R. de Luminy

Université d'Aix-Marseille

Rapport de recherche
sur le contrat
CRI n° 72-18 de
février 72 à juin 73

A Machine-Oriented Logic Based on the R

ARTIFICIAL INTELLIGENCE

Linear Resolution with Selectio

Donald Kı

En février 1972, le Groupe d'Intelligence Artificielle de
Luminy recevait une subvention de 180 000,00 francs dans le cadre
du contrat
langue nat
en juin 19

“. . . recei

“It can be int

Il peut
a été déper

- à pay

- à inv

- à pre
ligence art

The Semantics of Predicate Logic as a Program

M. H. VAN EMDEN AND R. A. KOWALSKI

University of Edinburgh, Edinburgh, Scotland

ABSTRACT Sentences in first-order predicate logic can be usefully interpreted as
operational and fixpoint semantics of predicate logic programs are defined, and th
theory and model theory of logic are investigated It is concluded that operationa
theory and that fixpoint semantics is a special case of model-theoretic semantics

AN ABSTRACT PROLOG INSTRUCTION SET

Technical Note 309

October 1983

By: David H.D. Warren, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

Alain Colmerauer (1941–2017)

- French computer scientist
- Natural language processing, PROLOG, constraint logic programming
- Knight of the French Legion of Honour (1986), AAAI Fellow (1991)

Robert Anthony Kowalski (b. 1941)

- American-British logician and computer scientist
- Logic programming, event calculus, abductive logic programming
- Doctoral advisor of David Warren, Keith Clark
- AAAI Fellow (1991), IJCAI Award for Research Excellence (2011)

Selection Function vs. Selection Rule

Recall

A program clause $A \leftarrow B_1, \ldots, B_n$ is a (definite) FOL clause $A \vee \neg B_1 \vee \ldots \vee \neg B_n$.

Definition

A **selection function** assigns to each non-empty clause C a literal $L \in C$.

Observation

- For a fact (unit clause) A, any selection function must select A.
- For a negated query $\neg(B_1, \ldots, B_n)$ (i.e. a clause $\neg B_1 \vee \ldots \vee \neg B_n$), any selection function must select a negative literal.
- For a program clause, a positive or a negative literal can be selected.

- Selecting a negative literal: Forward chaining (e.g. Datalog)
- Selecting the positive literal: Backward chaining (SLD resolution)

A **selection rule** restricts the selection function to (negated) queries.

FOL Resolution vs. SLD Resolution

Recall

For program P and query $B_1, \ldots, B_n$, we want to show $P \models B_1, \ldots, B_n$.

Observation

In first-order logic, $P \models B_1 \land \ldots \land B_n$ iff $P \cup \{\neg(B_1 \land \ldots \land B_n)\}$ is unsatisfiable.

- We use FOL resolution to show that $P \cup \{\neg B_1 \lor \ldots \lor \neg B_n\}$ is unsatisfiable.
- A backward-chaining selection function will always select positive literals from program clauses.
- So the only negative literals to resolve on can come from the (negated) query.
- Thus the ensuing resolution is linear in the sense that a (negated) query is involved in every step.

Turing-Completeness

Definite Clauses as Programming Language?

First-order clauses in combination with SLD resolution constitute a Turing-complete computation mechanism.

Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ can be cast as a logic program P_M:

- states $q \in Q$ represented by constants
- input/tape alphabet symbols $a \in \Gamma$ represented by unary functions
- words $w = a_1 a_2 \cdots a_n \in \Gamma^*$ represented as terms $t_w = a_1(a_2(\cdots a_n(e) \cdots))$
- thus the empty word ε is represented by the constant e
- tape content to the left of the head is in reverse: $t_w^R = a_n(a_{n-1}(\cdots a_1(e) \cdots))$
- configuration vqw of the TM represented by query $conf(t_v^R, q, t_w)$

Definite Clauses as Programming Language!

First-order clauses in combination with SLD resolution constitute a Turing-complete computation mechanism.

- transition function $\delta\colon Q \times \Gamma \to 2^{Q \times \Gamma \times \{l,n,r\}}$ expressed by clauses like

$$conf(V, q, a(W)) \leftarrow conf(b(V), s, W) \qquad \text{for each } (s, b, r) \in \delta(q, a)$$
$$conf(V, q, e) \leftarrow conf(b(V), s, e) \qquad \text{for each } (s, b, r) \in \delta(q, \square)$$

- acceptance is ensured via facts

$$conf(V, q, a(W)) \leftarrow \qquad \text{for each } q \in F, a \in \Gamma \text{ with } \delta(q, a) = \emptyset$$
$$conf(V, q, e) \leftarrow \qquad \text{for each } q \in F \text{ with } \delta(q, \square) = \emptyset$$

Theorem

TM M accepts w iff $P_M \cup \{conf(e, q_0, t_w)\}$ has a successful SLD derivation.

Conclusion

Summary

- Definite Horn clauses possess the **model intersection property**.
- Thus each definite logic program has a **unique least Herbrand model**.
- The least fixpoint of a program's **one-step consequence operator** T_P coincides with its least Herbrand model.
- First-order clauses in combination with SLD resolution constitute a **Turing-complete** computation mechanism.

Suggested action points:

- Find a (non-Horn) clause C with two Herbrand models I_1, I_2 where $I_1 \cap I_2 \not\models C$. (See Slide 6.)
- Show that T_P is monotonic.