TECHNISCHE UNIVERSITÄT DRESDEN

Computational Logic ∴ Group

Hannes Strass (based on slides by Michael Thielscher)

Faculty of Computer Science, Institute of Artificial Intelligence, Computational Logic Group

Least Herbrand Models

Lecture 5, 14th Nov 2022 // Foundations of Logic Programming, WS 2022/23

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 $A \in \bigcap \mathcal{K}$. □

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

Corollary

The set $\bigcap \{ I \mid I \text{ is a Herbrand model of P} \}$ 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 leaf	implies	$A \leftarrow \ \in \mathit{ground}(P)$
		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	$A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P)$ and $B_1, \ldots, B_n \in I$ (I.H.)
implies	$A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P)$ 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).

"$\Leftarrow$": Show for every interpretation I: $I \models P$ implies $I \models A$.

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

$I \models P$

implies $I \models A \leftarrow B_1, \ldots, B_n$ for all $A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P)$

implies if $I \models B_1, \ldots, I \models B_n$ then $I \models A$ for all ...

implies if $B_1 \in I_H, \ldots, B_n \in I_H$ then $A \in I_H$ for all ... (Def. I_H)

implies $I_H \models P$ (by Lemma 4.26; thus I_H is a Herbrand model)

implies $A \in I_H$ (since $A \in \mathcal{M}(P)$ and $\mathcal{M}(P)$ least Herbrand model)

implies $I \models A$ (by Def. I_H) □

Computing Least Herbrand Models

Complete Partial Orderings

Definition

Let $(\mathcal{A}, \sqsubseteq)$ be a partial ordering. (cf. Lecture 2)

- a **least** element of $X \subseteq \mathcal{A}$ $:\iff$ $a \in X$ and $a \sqsubseteq x$ for all $x \in X$
- a **least upper bound** of $X \subseteq \mathcal{A}$ (Notation: $a = \bigsqcup X$)
  $:\iff$ $a \in \mathcal{A}$, $x \sqsubseteq a$ for all $x \in X$,
  and a is the least element of $\mathcal{A}$ with this property

Definition

$(\mathcal{A}, \sqsubseteq)$ **complete** partial ordering (**cpo**) $:\iff$

- $\mathcal{A}$ 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 **monotonic** (or **order-preserving**)
  $:\iff$ for all $I_1, I_2 \in \mathcal{A}$: $I_1 \sqsubseteq I_2$ implies $T(I_1) \sqsubseteq T(I_2)$
- T **finitary** $:\iff$ 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 **continuous** $:\iff$ T 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 $I \in \mathcal{A}$.

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

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 **pre-fixpoint** of $T \;:\Longleftrightarrow\; T(I) \sqsubseteq I$
- I **fixpoint** of $T \;:\Longleftrightarrow\; T(I) = I$

Theorem 4.22 (Kleene's fixpoint theorem)

If T is a continuous operator on a CPO, 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.

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.

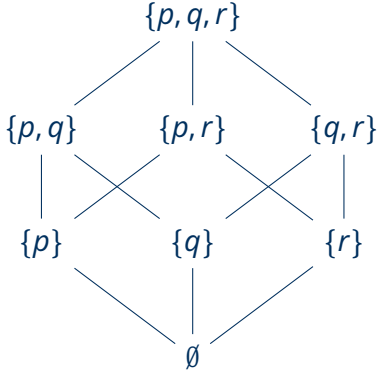
(ii) T_P is monotonic.

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 $P = \{p \leftarrow, \quad q \leftarrow p, \quad r \leftarrow r\}$.

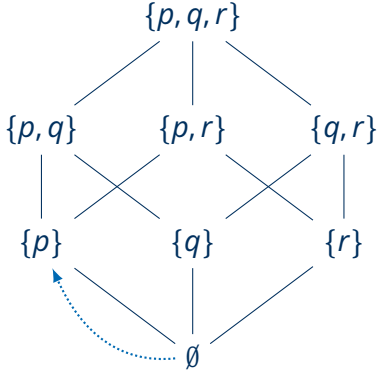
The operator T_P maps as follows:



T_P-Operator: Example (1)

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

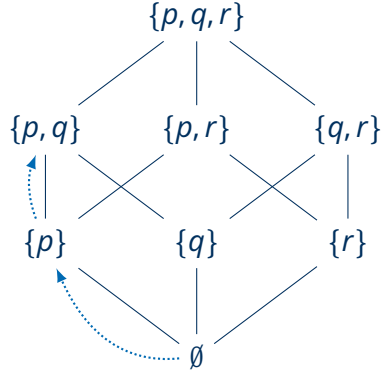
The operator T_P maps as follows:



T_P-Operator: Example (1)

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

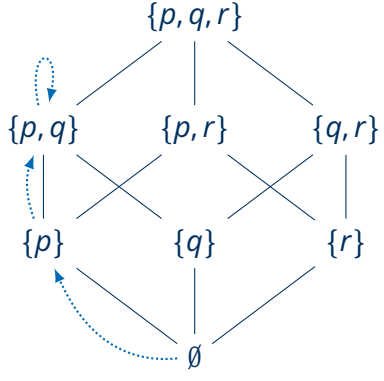
The operator T_P maps as follows:



T_P-Operator: Example (1)

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

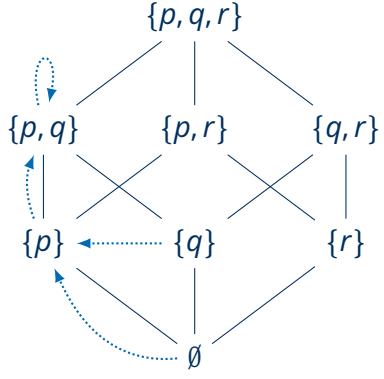
The operator T_P maps as follows:



T_P-Operator: Example (1)

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

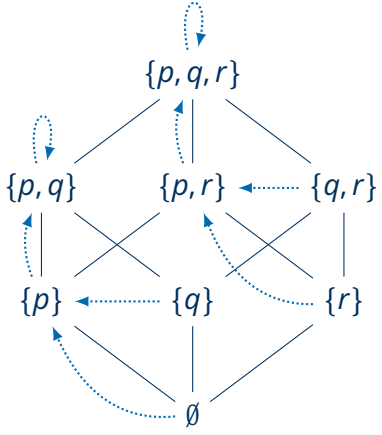
The operator T_P maps as follows:



T_P-Operator: Example (1)

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

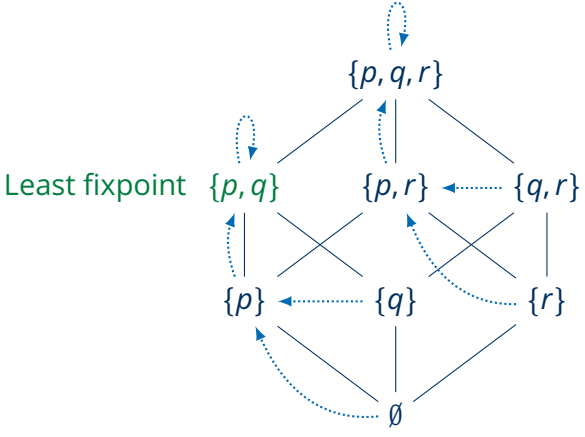
The operator T_P maps as follows:



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:



Quiz: T_P-Operator

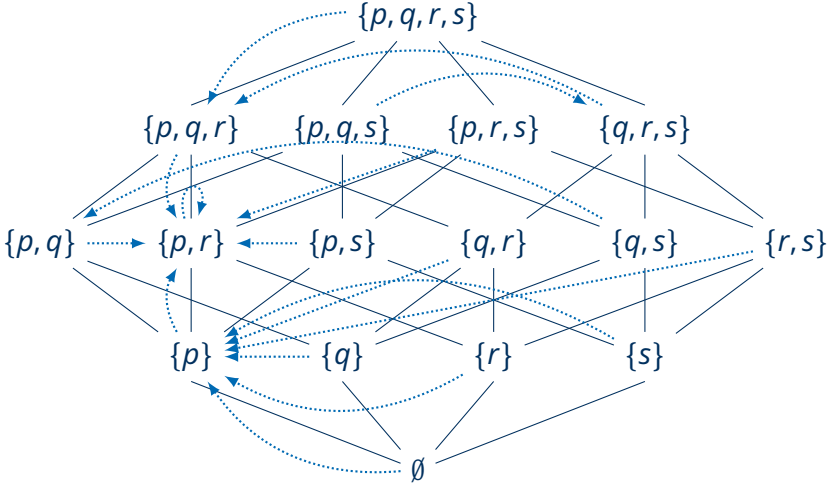
Recall: $T_P(I) := \{ A \mid A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P), \{B_1, \ldots, B_n\} \subseteq I \}$.

Quiz

Consider the following (definite) logic program: ...

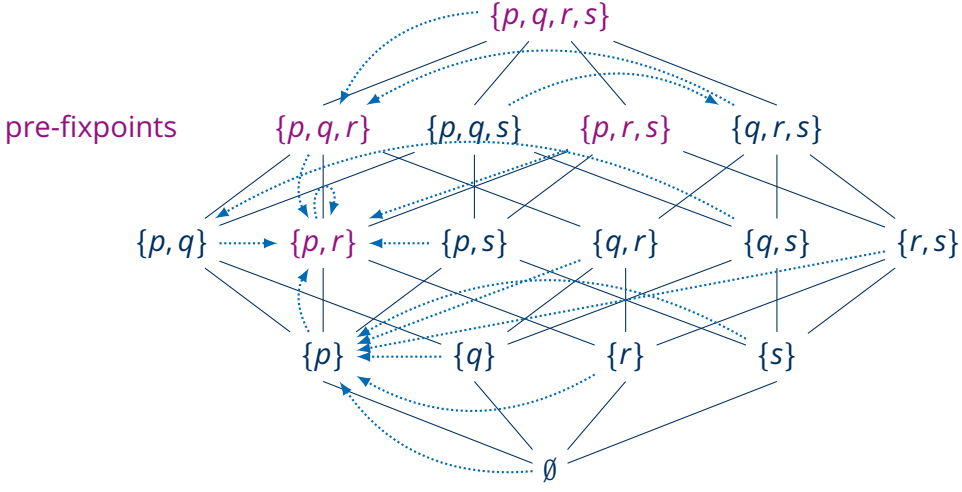
T_P-Operator: Example (2)

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



T_P-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.

$I \models P$
iff for every $A \leftarrow B_1, \ldots, B_n \in \mathit{ground}(P)$:
$\{B_1, \ldots, B_n\} \subseteq I$ implies $A \in I$ (by Lemma 4.26)
iff for every ground atom A: $A \in T_P(I)$ implies $A \in I$
iff $T_P(I) \subseteq I$ □

Characterization Theorem

Theorem 4.34

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

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) $P \models A$

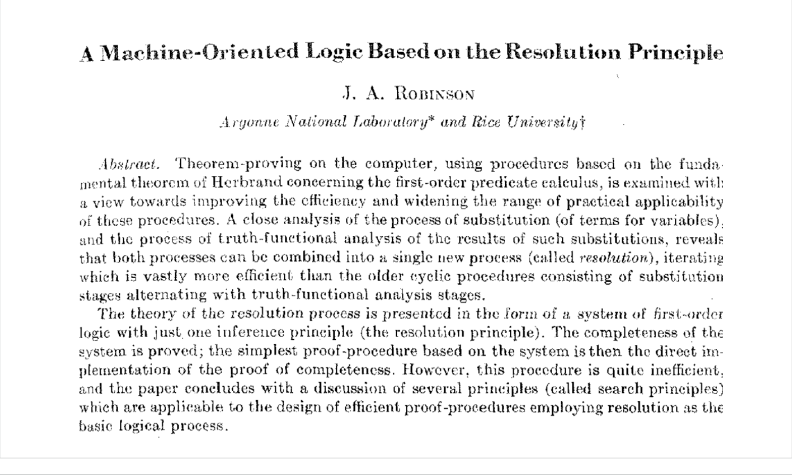
(iii) Every SLD tree for $P \cup \{A\}$ is successful

(iv) A is in the success set of P

History

Timeline

1965: John Alan Robinson: The resolution principle



A Machine-Oriented Logic Based on the Resolution Principle

J. A. Robinson

Argonne National Laboratory and Rice University†*

Abstract. Theorem-proving on the computer, using procedures based on the fundamental theorem of Herbrand concerning the first-order predicate calculus, is examined with a view towards improving the efficiency and widening the range of practical applicability of these procedures. A close analysis of the process of substitution (of terms for variables), and the process of truth-functional analysis of the results of such substitutions, reveals that both processes can be combined into a single new process (called *resolution*), iterating which is vastly more efficient than the older cyclic procedures consisting of substitution stages alternating with truth-functional analysis stages.

The theory of the resolution process is presented in the form of a system of first-order logic with just one inference principle (the resolution principle). The completeness of the system is proved; the simplest proof-procedure based on the system is then the direct implementation of the proof of completeness. However, this procedure is quite inefficient, and the paper concludes with a discussion of several principles (called search principles) which are applicable to the design of efficient proof-procedures employing resolution as the basic logical process.

Timeline

1965: John Alan Robinson: The resolution principle

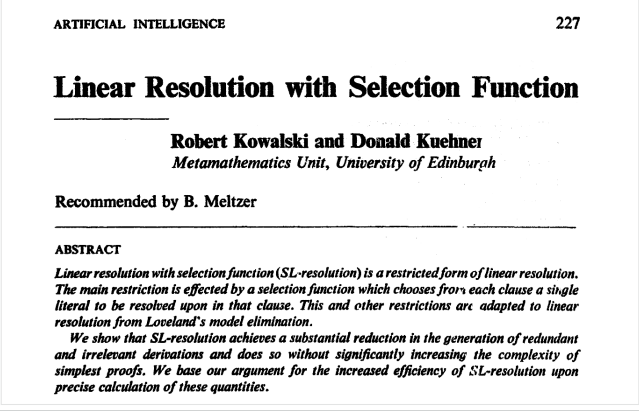
1970: Alain Colmerauer: Q-systems

Timeline

1965: John Alan Robinson: The resolution principle

1970: Alain Colmerauer: Q-systems

1971: Robert Kowalski & Donald Kuehner: SL-Resolution



ARTIFICIAL INTELLIGENCE 227

Linear Resolution with Selection Function

Robert Kowalski and Donald Kuehner
Metamathematics Unit, University of Edinburgh

Recommended by B. Meltzer

ABSTRACT

Linear resolution with selection function (SL-resolution) is a restricted form of linear resolution. The main restriction is effected by a selection function which chooses from each clause a single literal to be resolved upon in that clause. This and other restrictions are adapted to linear resolution from Loveland's model elimination.

We show that SL-resolution achieves a substantial reduction in the generation of redundant and irrelevant derivations and does so without significantly increasing the complexity of simplest proofs. We base our argument for the increased efficiency of SL-resolution upon precise calculation of these quantities.

Timeline

1965: John Alan Robinson: The resolution principle

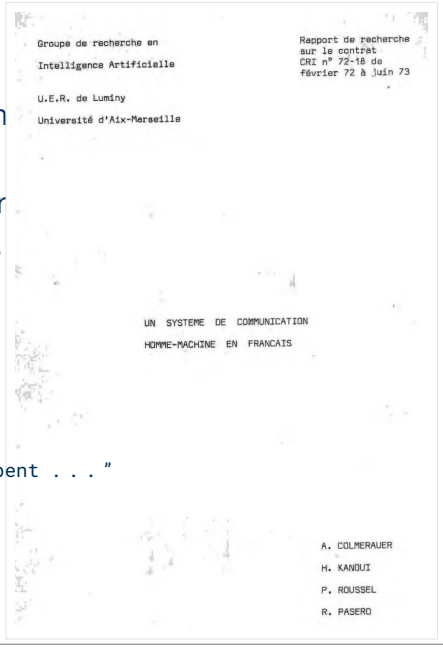
1970: Alain Colmerauer: Q-systems

1971: Robert Kowalski & Donald Kuehner: SL-Resolution

1971: Alain Colmerauer: Logic grammars

Timeline

1965: John Alan Robinson: The resolution
1970: Alain Colmerauer: Q-systems
1971: Robert Kowalski & Donald Kuehner
1971: Alain Colmerauer: Logic grammars
1972: Colmerauer et al.: PROLOG



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

UN SYSTEME DE COMMUNICATION
HOMME-MACHINE EN FRANCAIS

A. COLMERAUER
H. KANOUI
P. ROUSSEL
R. PASERO

En février 1972, le Groupe d'Intelligence Artificielle de Luminy recevait une subvention de 180 000,00 francs dans le cadre du contrat CRI 72-18 intitulé "Communication homme-machine en langue naturelle avec déduction automatique". Ce contrat se termina en juin 1973.

"... received a grant of 180,000 Francs ..."

"It can be interesting to know how the money has been spent ..."

Il peut être intéressant de savoir comment l'argent du contrat a été dépensé:

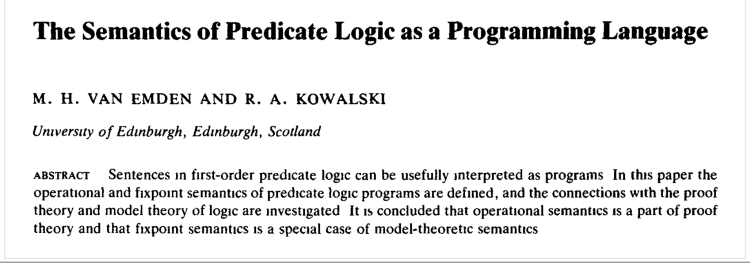
- à payer beaucoup d'heures machine

- à inviter R. KOWALSKI d'Edimbourg et J. TRUDEL de Montréal.

- à prendre contact avec les principaux laboratoires d'intelligence artificielle des Etats-Unis et d'Angleterre.

Timeline

1965: John Alan Robinson: The resolution principle
1970: Alain Colmerauer: Q-systems
1971: Robert Kowalski & Donald Kuehner: SL-Resolution
1971: Alain Colmerauer: Logic grammars
1972: Colmerauer et al.: PROLOG
1974: Robert Kowalski & Maarten van Emden: Least fixpoint semantics



The Semantics of Predicate Logic as a Programming Language

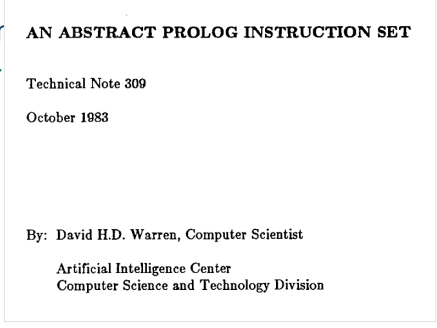
M. H. VAN EMDEN AND R. A. KOWALSKI

University of Edinburgh, Edinburgh, Scotland

ABSTRACT Sentences in first-order predicate logic can be usefully interpreted as programs In this paper the operational and fixpoint semantics of predicate logic programs are defined, and the connections with the proof theory and model theory of logic are investigated It is concluded that operational semantics is a part of proof theory and that fixpoint semantics is a special case of model-theoretic semantics

Timeline

1965: John Alan Robinson: The resolution principle
1970: Alain Colmerauer: Q-systems
1971: Robert Kowalski & Donald Kuehner: SL-Resolution
1971: Alain Colmerauer: Logic grammars
1972: Colmerauer et al.: PROLOG
1974: Robert Kowalski & Maarten van Emder
1983: David H. D. Warren: PROLOG compiler



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)

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)

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)

"Algorithm = Logic + Control"

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:

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 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.