Logic Programs
and Minimal Models

Marco Piastra

Logic Program

- An example of logic program:

$$\Pi \equiv \{\{Human(x), \neg Philosopher(x)\}, \{Mortal(y), \neg Human(y)\},$$
$$\{Philosopher(socrates)\}, \{Philosopher(plato)\}, \{Philosopher(aristotle)\}\}$$

$$\phi \equiv \exists x \, Mortal(x)$$

$$\neg\phi \equiv \neg\exists x \, Mortal(x)$$
$$\equiv \forall x \, \neg Mortal(x)$$
$$\equiv \{\neg Mortal(x)\} \qquad \text{(a goal, i.e. a Horn clause)}$$

By applying resolution in an exhaustive way, we obtain:

$$\Sigma \equiv \{[x/socrates], [x/plato], [x/aristotle]\}$$

Looks like a query on an *implicit* database …

- **Answer Set**

    It includes all complete substitutions of the variables in the *goal*
    corresponding to the closed branches (i.e. with an empty clause) in the SLD tree

Herbrand Universe, Herbrand Base

- ## Herbrand **terms** and **atoms**

  Given a signature Σ

  A Herbrand **term** is a *ground term* (i.e. a term that contains no variables)

  Examples:

  $f(a)$, $g(a,b)$, $g(f(a),b)$, $g(f(a),g(b,c))$, $g(f(a),g(f(b),c))$, …

  A Herbrand **atom** is a *ground atom* (i.e. an atom that contains no variables)

  Examples:

  $P(f(a))$, $P(g(a,b))$, $Q(g(f(a),b)$, $g(f(a),g(b,c)))$, …

- ## Herbrand **universe**

  The set of all Herbrand *terms* from Σ

  Example:

  $\mathbf{U}_H \equiv \{f(a), g(a,b), g(f(a),b), g(f(a),g(b,c)), g(f(a),g(f(b),c)), … \}$

- ## Herbrand **base**

  The set of all Herbrand *atoms* from Σ

  Example:

  $\mathbf{B}_H \equiv \{P(f(a)), P(g(a,b)), Q(g(f(a),b), g(f(a),g(b,c))), …\}$

Herbrand models

- ## Herbrand **structure**

  A semantic structure $<\mathbf{U}_H, \Sigma, v_H>$ such that

- ## Herbrand **interpretation** v_H

  For constants, $v_H(c) = c$

  For ground terms, $v_H(t) = t$

  For predicate symbols, $v_H \subseteq \mathbf{B}_H$

    i.e. a **subset** of the Herbrand base $\mathbf{B}_H$

    Example: $v_H \equiv \{P(a), P(f(b)), P(c), Q(a,g(b,c)), Q(b,c) \dots \}$

- ## Herbrand **model**

  $\varphi \in \text{Atom}(L_{PO}),\ <\mathbf{U}_H, \Sigma, v_H>[s] \models \varphi$ iff $\varphi \in v_H$

  $\varphi \in \text{Atom}(L_{PO}),\ <\mathbf{U}_H, \Sigma, v_H>[s] \models \neg\varphi$ iff $\varphi \notin v_H$

  $<\mathbf{U}_H, \Sigma, v_H>[s] \models \neg\varphi$ iff $<\mathbf{U}_H, \Sigma, v_H>[s] \not\models \varphi$

  $<\mathbf{U}_H, \Sigma, v_H>[s] \models \varphi \to \psi$ iff $(<\mathbf{U}_H, \Sigma, v_H>[s] \not\models \varphi$ or $<\mathbf{U}_H, \Sigma, v_H>[s] \models \psi)$

  $<\mathbf{U}_H, \Sigma, v_H>[s] \models \forall x\, \varphi$ iff for all $c \in \text{Cost}(L_{PO})$, $<\mathbf{U}_H, \Sigma, v_H>[s](x{:}c) \models \varphi$

Horn clauses and Herbrand models

- **Herbrand Theorem**

  Given a theory of universal sentences Φ,
  $H(\Phi)$ has a model iff Φ has a model

- Corollary (for *Horn clauses*)

  Given a set Φ of <u>Horn clauses</u>, the two following statements are equivalent:
  - Φ is satisfiable
  - Φ has an <u>Herbrand model</u>

    <u>This is not true in general</u>: only if Φ is a set of Horn clauses

Clausole di Horn e modelli di Herbrand

- Corollary to Herbrand theorem (for *Horn clauses*)

  Given a set Φ of <u>Horn clauses</u>, the two following statements are equivalent:
  - Φ is satisfiable
  - Φ has an <u>Herbrand model</u>

  <u>This is not true in general</u>: only if Φ is a set of Horn clauses

- Herbrand **minimal model**

  The minimal model M_Φ for a set of Horn clauses Φ is:

  $$M_\Phi \equiv \cap_{\forall i} M_i \qquad \text{where } M_i \text{ is a Herbrand model of } \Phi$$

- Theorem(van Emden e Kowalski, 1976)

  Let Φ be a set of Horn clauses and φ a <u>ground</u> atom.
  These three statements are equivalent:
  - $\Phi \models \varphi$
  - $\varphi \in M_\Phi$
  - φ is derivable from Φ via resolution with refutation

Logic programming system and minimal model

- **Theorem** (Apt e van Emden, 1982)

  Let Π be a ***logical program*** (i.e. *a set of definite clauses*).
  The (finite) success set of Π with SLD-resolution (*fair*) coincides with M_Π

- A logic programming system (i.e. Prolog) can generate the *subset* of M_Π corresponding to a specific ***goal***

  A goal $\{\neg\alpha_1, \neg\alpha_2, ..., \neg\alpha_m\}$ where the variables $x_1, x_2, ..., x_m$ occur
  is equivalent to the sentence $\forall x_1 \forall x_2 \ldots \forall x_n (\neg\alpha_1 \lor \neg\alpha_2 \lor ... \lor \neg\alpha_m)$
  which is equivalent to $\neg\exists x_1 \exists x_2 \ldots \exists x_n (\alpha_1 \land \alpha_2 \land ... \land \alpha_m)$

  A logic programming system can generate all possible ***substitutions*** $[x_1/t_1, x_2/t_2, ..., x_n/t_n]$
  such that $\Pi \cup \{\neg(\alpha_1 \land \alpha_2 \land ... \land \alpha_m)[x_1/t_1, x_2/t_2, ..., x_n/t_n]\}$ is *unsatisfiable*

  (that implies $\Pi \models (\alpha_1 \land \alpha_2 \land ... \land \alpha_m)[x_1/t_1, x_2/t_2, ..., x_n/t_n])$

  (that implies $(\alpha_1 \land \alpha_2 \land ... \land \alpha_m)[x_1/t_1, x_2/t_2, ..., x_n/t_n] \in M_\Pi)$

  Each goal act like a *filter*, i.e. defining the subset of M_Π

  *NOTE: a logic programming system with a **fair** strategy can do so...*