Introduction to Answer Set Programming (ASP): Logic Programming and Non-Monotonic Reasoning

Jianmin Ji and Guoqiang Jin

{jizheng,abxeeled}@mail.ustc.edu.cn
Multi-Agent Systems Lab.
School of Computer Science and Technology
University of Science and Technology of China

August 17, 2010

Outline

1. 关于暑期讨论班
2. 逻辑程序
3. 稳定模型
4. 相关阅读材料

Outline

1. 关于暑期讨论班
2. 逻辑程序
3. 稳定模型
4. 相关阅读材料

机器人智能决策讨论班 (2010 年暑假)

- 主要内容：
  - Answer Set Programming (ASP) 基础：历史背景，语法，语义，基本性质。
  - ASP 求解：计算原理，多种求解器介绍，使用和比较。
  - ASP 编程：使用 ASP 来求解问题，Home 组仿真平台的熟悉以及编程。
  - 自然语言理解，以及自然语言理解在 Home 机器人决策中的应用。
  - 行动推理的理论以及相关的实验。

机器人智能决策讨论班 (2010 年暑假)

- 主要内容：
  - Answer Set Programming (ASP) 基础：历史背景，语法，语义，基本性质。
  - ASP 求解：计算原理，多种求解器介绍，使用和比较。
  - ASP 编程：使用 ASP 来求解问题，Home 组仿真平台的熟悉以及编程。
  - 自然语言理解，以及自然语言理解在 Home 机器人决策中的应用。
  - 行动推理的理论以及相关的实验。
- 方式：
  - 报告：隔一天一次专题报告。见 `http://wrighteagle.org/cn/seminar/theory_10.php`
  - 文献阅读：在报告之前和之后，有一些文献需要阅读。
  - 讨论：每次报告后有一段时间集体讨论或回答问题。
  - 实验：主要是 Home 组的仿真平台。

ASP 基础与应用讨论班 (2010 年暑假) con't

期望效果

- 了解 ASP 理论基础。
- 掌握 ASP 编程技术，可以用来求解实际问题。
- 了解行动推理或自然语言理解的大致框架。
- 了解并实验 Home 机器人中使用的 ASP 智能决策技术。

Outline

1. 关于暑期讨论班
2. 逻辑程序
3. 稳定模型
4. 相关阅读材料

逻辑程序

- Answer Set Programming (ASP) 是逻辑程序 (Logic Programming) 中的一种。

逻辑程序

- Answer Set Programming (ASP) 是逻辑程序 (Logic Programming) 中的一种。
- 逻辑程序的基本观点：Algorithm = Logic + Control

逻辑程序

- Answer Set Programming (ASP) 是逻辑程序 (Logic Programming) 中的一种。
- 逻辑程序的基本观点：Algorithm = Logic + Control
- 只需用逻辑程序将问题的逻辑部分 (问题是什么) 描述清楚，程序自动求解。

逻辑程序

- Answer Set Programming (ASP) 是逻辑程序 (Logic Programming) 中的一种。
- 逻辑程序的基本观点：Algorithm = Logic + Control
- 只需用逻辑程序将问题的逻辑部分 (问题是什么) 描述清楚，程序自动求解。
- 在逻辑程序中引入 'not'，从而处理非单调推理。

逻辑程序的语法

- 在逻辑程序中 (正程序)，所有“语句”都表示成“规则”的形式：

$$A \leftarrow A_1, A_2, \ldots, A_m.$$

其中 A 是一个原子，称为该规则的“头”， A_i 也是原子，合称为该规则的“体”。

逻辑程序的语法

- 在逻辑程序中 (正程序)，所有“语句”都表示成“规则”的形式：

$$A \leftarrow A_1, A_2, \ldots, A_m.$$

  其中 A 是一个原子，称为该规则的“头”， A_i 也是原子，合称为该规则的“体”。
- $n = 0$ 的规则称为“事实”，没有体，只有头，往往省略反箭头 $\leftarrow$.

逻辑程序的语法

- 在逻辑程序中 (正程序)，所有“语句”都表示成“规则”的形式：

$$A \leftarrow A_1, A_2, \ldots, A_m.$$

  其中 A 是一个原子，称为该规则的“头”，A_i 也是原子，合称为该规则的“体”。
- $n = 0$ 的规则称为“事实”，没有体，只有头，往往省略反箭头 $\leftarrow$.
- 文字的析取称为一个“子句”。不严格的说，一条规则 $A \leftarrow A_1, A_2, \ldots, A_m$ 代表一个子句 $A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$。

逻辑程序的语法

- 在逻辑程序中 (正程序)，所有“语句”都表示成“规则”的形式：

$$A \leftarrow A_1, A_2, \ldots, A_m.$$

  其中 A 是一个原子，称为该规则的“头”，A_i 也是原子，合称为该规则的“体”。
- $n = 0$ 的规则称为“事实”，没有体，只有头，往往省略反箭头 $\leftarrow$.
- 文字的析取称为一个“子句”。不严格的说，一条规则 $A \leftarrow A_1, A_2, \ldots, A_m$ 代表一个子句 $A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$。
- 仅有一个正文字的子句，称为 Horn 子句。每条逻辑程序 (正程序) 的规则对应一个 Horn 子句。

逻辑程序的语法

- 在逻辑程序中 (正程序)，所有“语句”都表示成“规则”的形式：

$$A \leftarrow A_1, A_2, \ldots, A_m.$$

  其中 A 是一个原子，称为该规则的“头”， A_i 也是原子，合称为该规则的“体”。
- $n = 0$ 的规则称为“事实”，没有体，只有头，往往省略反箭头 $\leftarrow$.
- 文字的析取称为一个“子句”。不严格的说，一条规则 $A \leftarrow A_1, A_2, \ldots, A_m$ 代表一个子句 $A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$。
- 仅有一个正文字的子句，称为 Horn 子句。每条逻辑程序 (正程序) 的规则对应一个 Horn 子句。
- 默认子句中出现的所有个体变元都被全称量化。因此，子句 $A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$ 实际上是全称闭式：

$$\forall x_1, x_2, \ldots, x_n : A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$$

  的简写，其中 $x_1, x_2, \ldots, x_n$ 是子句中出现的所有个体变元。

Herbrand 基

- 任给程序 P, P 的一个“常项”(ground term) 是一个由 P 中出现的个体常元和函数符号复合而成的项。

Herbrand 基

- 任给程序 P, P 的一个 "常项" (ground term) 是一个由 P 中出现的个体常元和函数符号复合而成的项。
- P 的所有常项的集合称为 P 的 Herbrand 域 (Herbrand Universe)，记为 $U(P)$。

Herbrand 基

- 任给程序 P, P 的一个“常项”(ground term) 是一个由 P 中出现的个体常元和函数符号复合而成的项。
- P 的所有常项的集合称为 P 的 Herbrand 域 (Herbrand Universe)，记为 $U(P)$。
- 例如： $P_1 = \{ p(1)., q(2)., q(x) \leftarrow p(x). \}$，则 $U(P_1) = \{ 1, 2 \}$。

Herbrand 基

- 任给程序 P, P 的一个“常项”(ground term) 是一个由 P 中出现的个体常元和函数符号复合而成的项。
- P 的所有常项的集合称为 P 的 Herbrand 域 (Herbrand Universe)，记为 $U(P)$。
- 例如： $P_1 = \{ p(1)., q(2)., q(x) \leftarrow p(x). \}$，则 $U(P_1) = \{ 1, 2 \}$。
- P 的一个“常原子”是一个由 P 的常项和 P 中出现的谓词符号组成的原子。

Herbrand 基

- 任给程序 P, P 的一个“常项”(ground term) 是一个由 P 中出现的个体常元和函数符号复合而成的项。
- P 的所有常项的集合称为 P 的 Herbrand 域 (Herbrand Universe)，记为 $U(P)$。
- 例如： $P_1 = \{ p(1).,q(2).,q(x) \leftarrow p(x). \}$，则 $U(P_1) = \{ 1,2 \}$。
- P 的一个“常原子”是一个由 P 的常项和 P 中出现的谓词符号组成的原子。
- P 的所有常原子的集合称为 P 的 Herbrand 基 (Herbrand Base)，记为 $B(P)$。例如， $B(P_1) = \{ p(1),p(2),q(1),q(2) \}$.

逻辑程序的常例 (grounding)

- 任给一个规则 R，R 的一个“常例”(ground instance) 是通过下述操作产生的不含个体变元的规则：对 R 中出现的每一个体变元 x，处处同时带入一个常项。

逻辑程序的常例 (grounding)

- 任给一个规则 R，R 的一个“常例”(ground instance) 是通过下述操作产生的不含个体变元的规则：对 R 中出现的每一个体变元 x，处处同时带入一个常项。
- 例如，P_1 中第三条规则有两个常例:

$$q(1) \leftarrow p(1).$$
$$q(2) \leftarrow p(2).$$

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。
- 一阶逻辑语义：一个解释 I 由一个论域 D 以及一个从项和公式到 D 上函数和关系的映射 V 构成。给定 I，任何公式都可被映射为真或者假。

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。
- 一阶逻辑语义：一个解释 I 由一个论域 D 以及一个从项和公式到 D 上函数和关系的映射 V 构成。给定 I，任何公式都可被映射为真或者假。
- 任给程序 P， P 的任何一个 Herbrand 解释 I_H 的论域取为 $U(P)$，即 P 的 Herbrand 域。 I_H 的解释函数 V_H 将 P 的任意常项和常原子映射为它们自身。

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。
- 一阶逻辑语义：一个解释 I 由一个论域 D 以及一个从项和公式到 D 上函数和关系的映射 V 构成。给定 I，任何公式都可被映射为真或者假。
- 任给程序 P， P 的任何一个 Herbrand 解释 I_H 的论域取为 $U(P)$，即 P 的 Herbrand 域。 I_H 的解释函数 V_H 将 P 的任意常项和常原子映射为它们自身。
- P 的一个 Herbrand 解释还包含一个由 P 的若干常原子组成的“指派”集合 $M(P) \subseteq B(P)$。一个常原子 p 在指派 $M(P)$ 下为真，当且仅当 $p \in M(P)$。

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。
- 一阶逻辑语义：一个解释 I 由一个论域 D 以及一个从项和公式到 D 上函数和关系的映射 V 构成。给定 I，任何公式都可被映射为真或者假。
- 任给程序 P， P 的任何一个 Herbrand 解释 I_H 的论域取为 $U(P)$，即 P 的 Herbrand 域。 I_H 的解释函数 V_H 将 P 的任意常项和常原子映射为它们自身。
- P 的一个 Herbrand 解释还包含一个由 P 的若干常原子组成的“指派”集合 $M(P) \subseteq B(P)$。一个常原子 p 在指派 $M(P)$ 下为真，当且仅当 $p \in M(P)$。
- 任何规则 R 在一个 Herbrand 解释 I_H 下的真值根据 $M(P)$ 如下：

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。
- 一阶逻辑语义：一个解释 I 由一个论域 D 以及一个从项和公式到 D 上函数和关系的映射 V 构成。给定 I，任何公式都可被映射为真或者假。
- 任给程序 P， P 的任何一个 Herbrand 解释 I_H 的论域取为 $U(P)$，即 P 的 Herbrand 域。 I_H 的解释函数 V_H 将 P 的任意常项和常原子映射为它们自身。
- P 的一个 Herbrand 解释还包含一个由 P 的若干常原子组成的“指派”集合 $M(P) \subseteq B(P)$。一个常原子 p 在指派 $M(P)$ 下为真，当且仅当 $p \in M(P)$。
- 任何规则 R 在一个 Herbrand 解释 I_H 下的真值根据 $M(P)$ 如下：
    1. $I_H(R) = t$，当且仅当对 R 的所有常例 R^* 有 $I_H(R^*) = t$;

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。
- 一阶逻辑语义：一个解释 I 由一个论域 D 以及一个从项和公式到 D 上函数和关系的映射 V 构成。给定 I，任何公式都可被映射为真或者假。
- 任给程序 P， P 的任何一个 Herbrand 解释 I_H 的论域取为 $U(P)$，即 P 的 Herbrand 域。 I_H 的解释函数 V_H 将 P 的任意常项和常原子映射为它们自身。
- P 的一个 Herbrand 解释还包含一个由 P 的若干常原子组成的“指派”集合 $M(P) \subseteq B(P)$。一个常原子 p 在指派 $M(P)$ 下为真，当且仅当 $p \in M(P)$。
- 任何规则 R 在一个 Herbrand 解释 I_H 下的真值根据 $M(P)$ 如下：
  1. $I_H(R) = t$，当且仅当对 R 的所有常例 R^* 有 $I_H(R^*) = t$；
  2. 若 $A \leftarrow A_1, A_2, \ldots, A_m$ 是一个规则的常例，则

$$I_H(A \leftarrow A_1, A_2, \ldots, A_m) =_{df} I_H(A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m),$$

当且仅当， $I_H(A) = t$ 或者存在 A_i 使得 $I_H(\neg A_i) = t$，
当且仅当， $A \in M(P)$ 或者存在 $A_i \notin M(P)$。

逻辑程序的 Herbrand 模型

- 一个 Herbrand 解释称为一个规则的 Herbrand 模型，如果该规则在该解释下为真。

逻辑程序的 Herbrand 模型

- 一个 Herbrand 解释称为一个规则的 Herbrand 模型，如果该规则在该解释下为真。
- 一个 Herbrand 解释称为一个程序的 Herbrand 模型，如果该程序的所有规则在该解释下为真。

逻辑程序的 Herbrand 模型

- 一个 Herbrand 解释称为一个规则的 Herbrand 模型，如果该规则在该解释下为真。
- 一个 Herbrand 解释称为一个程序的 Herbrand 模型，如果该程序的所有规则在该解释下为真。
- 严格说，一个程序 P 的一个 Herbrand 解释是一个三元组 $I_H = \langle U(P), V_H, M(P) \rangle$。但 $U(P)$ 和 V_H 对任何 P 都是“固定的”，所以通常用指派集合 $M(P) \subseteq B(P)$ 代表一个 Herbrand 模型，简记为 M。

逻辑程序的 Herbrand 模型

- 一个 Herbrand 解释称为一个规则的 Herbrand 模型，如果该规则在该解释下为真。
- 一个 Herbrand 解释称为一个程序的 Herbrand 模型，如果该程序的所有规则在该解释下为真。
- 严格说，一个程序 P 的一个 Herbrand 解释是一个三元组 $I_H = \langle U(P), V_H, M(P) \rangle$。但 $U(P)$ 和 V_H 对任何 P 都是"固定的"，所以通常用指派集合 $M(P) \subseteq B(P)$ 代表一个 Herbrand 模型，简记为 M。
- 例如， $P_1 = \{ p(1)., q(2)., q(x) \leftarrow p(x). \}$ 有两个 Herbrand 模型: $\{ p(1), q(1), q(2) \}$ 和 $\{ p(1), p(2), q(1), q(2) \}$。

正程序 Herbrand 模型性质

Proposition 1

任何正程序都有 *Herbrand* 模型。

正程序 Herbrand 模型性质

Proposition 1

任何正程序都有 *Herbrand* 模型。

Theorem 1

正程序的任意两个 *Herbrand* 模型的交集也是该程序的一个 *Herbrand* 模型。

正程序 Herbrand 模型性质

Proposition 1

任何正程序都有 *Herbrand* 模型。

Theorem 1

正程序的任意两个 *Herbrand* 模型的交集也是该程序的一个 *Herbrand* 模型。

Proposition 2

任何正程序有唯一的极小（最小）*Herbrand* 模型。记为 $AS(P)$。

正程序 Herbrand 模型性质

Proposition 1

任何正程序都有 *Herbrand* 模型。

Theorem 1

正程序的任意两个 *Herbrand* 模型的交集也是该程序的一个 *Herbrand* 模型。

Proposition 2

任何正程序有唯一的极小（最小）*Herbrand* 模型。记为 $AS(P)$。

Proposition 3

两个逻辑程序 P_1, P_2，如果 $P_1 \subseteq P_2$，则 $AS(P_1) \subseteq AS(P_2)$。

Outline

1. 关于暑期讨论班
2. 逻辑程序
3. **稳定模型**
4. 相关阅读材料

逻辑程序中引入 ‘not’

- 逻辑程序中规则定义为如下形式：

$$A \leftarrow A_1, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n.$$

逻辑程序中引入 ‘not’

- 逻辑程序中规则定义为如下形式：

$$A \leftarrow A_1, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n.$$

- 引入 ‘not’，从而引入非单调的表达能力。

逻辑程序中引入 ‘not’

- 逻辑程序中规则定义为如下形式：

$$A \leftarrow A_1, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n.$$

- 引入 ‘not’，从而引入非单调的表达能力。
- ‘not’ 表示 “推不出”，逻辑程序下的推出 (推不出) 关系。

逻辑程序中引入 ‘not’

- 逻辑程序中规则定义为如下形式：

$$A \leftarrow A_1, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n.$$

- 引入 ‘not’，从而引入非单调的表达能力。
- ‘not’ 表示“推不出”，逻辑程序下的推出 (推不出) 关系。
- 对于正程序有唯一的极小 Herbrand 模型，对于一般逻辑程序极小 Herbrand 模型不是唯一的。(哪些才是合理的？)

稳定模型语义 (Stable Model Semantics)

The stable model semantics for logic programming [Gelfond & Lifschitz 1988].

- Assumption = Minimal Knowledge.

稳定模型语义 (Stable Model Semantics)

The stable model semantics for logic programming [Gelfond & Lifschitz 1988].

- Assumption = Minimal Knowledge.
- 对于含 'not' 的规则，我们不得不预先合理的假定一些知识 (assumption)。所谓的合理在这里解释为：与最后推导出的知识 (minimal knowledge) 一致。

稳定模型语义 (Stable Model Semantics)

The stable model semantics for logic programming [Gelfond & Lifschitz 1988].

- Assumption = Minimal Knowledge.
- 对于含 'not' 的规则，我们不得不预先合理的假定一些知识 (assumption)。所谓的合理在这里解释为：与最后推导出的知识 (minimal knowledge) 一致。
- 对于任意原子的集合 S， P^S 定义为 P 中按下列方式处理得到的程序：

稳定模型语义 (Stable Model Semantics)

The stable model semantics for logic programming [Gelfond & Lifschitz 1988].

- Assumption = Minimal Knowledge.
- 对于含 'not' 的规则，我们不得不预先合理的假定一些知识 (assumption)。所谓的合理在这里解释为：与最后推导出的知识 (minimal knowledge) 一致。
- 对于任意原子的集合 S， P^S 定义为 P 中按下列方式处理得到的程序：
  1. 如果规则体中有 $not\, a$，并且 $a \in S$，则删除这条规则；

稳定模型语义 (Stable Model Semantics)

The stable model semantics for logic programming [Gelfond & Lifschitz 1988].

- Assumption = Minimal Knowledge.
- 对于含 'not' 的规则，我们不得不预先合理的假定一些知识 (assumption)。所谓的合理在这里解释为：与最后推导出的知识 (minimal knowledge) 一致。
- 对于任意原子的集合 S， P^S 定义为 P 中按下列方式处理得到的程序：
  1. 如果规则体中有 $not\, a$，并且 $a \in S$，则删除这条规则；
  2. 删除剩余的含 not 文字。

稳定模型语义 (Stable Model Semantics)

The stable model semantics for logic programming [Gelfond & Lifschitz 1988].

- Assumption = Minimal Knowledge.
- 对于含 'not' 的规则，我们不得不预先合理的假定一些知识 (assumption)。所谓的合理在这里解释为：与最后推导出的知识 (minimal knowledge) 一致。
- 对于任意原子的集合 S， P^S 定义为 P 中按下列方式处理得到的程序：
  1. 如果规则体中有 $not\, a$，并且 $a \in S$，则删除这条规则；
  2. 删除剩余的含 not 文字。
- S 是 P 的稳定模型 iff S 是 P^S 的极小模型。

$$S = AS(P^S)$$

Stable Model 性质

Theorem 2

S 是程序 P 的 *Herbrand* 模型当且仅当 S 是 P^S 的 *Herbrand* 模型。

Stable Model 性质

Theorem 2

S 是程序 P 的 *Herbrand* 模型当且仅当 S 是 P^S 的 *Herbrand* 模型。

Theorem 3

程序 P 的任何一个稳定模型都是 P 的一个极小 *Herbrand* 模型。

Stable Model 性质

Theorem 2

S 是程序 P 的 *Herbrand* 模型当且仅当 S 是 P^S 的 *Herbrand* 模型。

Theorem 3

程序 P 的任何一个稳定模型都是 P 的一个极小 *Herbrand* 模型。

- 正程序有唯一的稳定模型，即该程序的极小模型；

Stable Model 性质

Theorem 2

S 是程序 P 的 *Herbrand* 模型当且仅当 S 是 P^S 的 *Herbrand* 模型。

Theorem 3

程序 P 的任何一个稳定模型都是 P 的一个极小 *Herbrand* 模型。

- 正程序有唯一的稳定模型，即该程序的极小模型；
- 有些程序没有稳定模型，例如 $\{\, p \leftarrow not\, p.\, \}$；

Stable Model 性质

Theorem 2

S 是程序 P 的 *Herbrand* 模型当且仅当 S 是 P^S 的 *Herbrand* 模型。

Theorem 3

程序 P 的任何一个稳定模型都是 P 的一个极小 *Herbrand* 模型。

- 正程序有唯一的稳定模型，即该程序的极小模型；
- 有些程序没有稳定模型，例如 $\{p \leftarrow not\, p.\}$；
- 有些程序有多个稳定模型，例如 $\{p \leftarrow not\, q.\ q \leftarrow not\, p.\}$；

Stable Model 性质

Theorem 2

S 是程序 P 的 *Herbrand* 模型当且仅当 S 是 P^S 的 *Herbrand* 模型。

Theorem 3

程序 P 的任何一个稳定模型都是 P 的一个极小 *Herbrand* 模型。

- 正程序有唯一的稳定模型，即该程序的极小模型;
- 有些程序没有稳定模型，例如 $\{\, p \leftarrow not\, p.\,\}$;
- 有些程序有多个稳定模型，例如 $\{\, p \leftarrow not\, q.\ q \leftarrow not\, p.\,\}$;
- 有些程序只有空集作为稳定模型，例如 $\{\, p \leftarrow p.\,\}$;

Stable Model 性质

Theorem 2

S 是程序 P 的 *Herbrand* 模型当且仅当 S 是 P^S 的 *Herbrand* 模型。

Theorem 3

程序 P 的任何一个稳定模型都是 P 的一个极小 *Herbrand* 模型。

- 正程序有唯一的稳定模型，即该程序的极小模型；
- 有些程序没有稳定模型，例如 $\{ p \leftarrow not\, p. \}$;
- 有些程序有多个稳定模型，例如 $\{ p \leftarrow not\, q.\ q \leftarrow not\, p. \}$;
- 有些程序只有空集作为稳定模型，例如 $\{ p \leftarrow p. \}$;
- 一个程序的稳定模型彼此互不为子集。

稳定模型语义举例

企鹅与鸟的例子:

稳定模型语义举例

企鹅与鸟的例子:

Example 4

$fly(X) \leftarrow bird(X), not\, nfly(X).$ /* 通常，鸟会飞 */

$bird(X) \leftarrow penguin(X).$ /* 企鹅是鸟 */

$nfly(X) \leftarrow penguin(X), not\, fly(X).$ /* 通常，企鹅不会飞 */

$penguin(tweety).$ /* tweety 是企鹅 */

稳定模型语义举例

企鹅与鸟的例子:

Example 4

$$fly(X) \leftarrow bird(X), not\, nfly(X).$$ /* 通常，鸟会飞 */

$$bird(X) \leftarrow penguin(X).$$ /* 企鹅是鸟 */

$$nfly(X) \leftarrow penguin(X), not\, fly(X).$$ /* 通常，企鹅不会飞 */

$$penguin(tweety).$$ /* tweety 是企鹅 */

tweety 会不会飞?

稳定模型语义举例

企鹅与鸟的例子:

Example 4

$$fly(X) \leftarrow bird(X), not\, nfly(X). \qquad \text{/* 通常，鸟会飞 */}$$
$$bird(X) \leftarrow penguin(X). \qquad \text{/* 企鹅是鸟 */}$$
$$nfly(X) \leftarrow penguin(X), not\, fly(X). \qquad \text{/* 通常，企鹅不会飞 */}$$
$$penguin(tweety). \qquad \text{/* tweety 是企鹅 */}$$

tweety 会不会飞?
此程序的所有稳定模型为：

$$\{ bird(tweety), penguin(tweety), fly(tweety) \},$$
$$\{ bird(tweety), penguin(tweety), nfly(tweety) \}.$$

经典否定

- 前面的介绍中，没有涉及经典否定 ¬。在程序中其实可以加入经典否定:

$$L \leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n.$$

其中， L 和 L_i 是文字，即原子或原子的否定 (¬)。

经典否定

- 前面的介绍中，没有涉及经典否定 ¬。在程序中其实可以加入经典否定:

$$L \leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n.$$

  其中， L 和 L_i 是文字，即原子或原子的否定 (¬)。
- 对于含经典否定的正逻辑程序，仍然可以定义极小 Herbrand 模型，只不过此时 Herbrand 解释对应为文字的集合。

经典否定

- 前面的介绍中，没有涉及经典否定 ¬。在程序中其实可以加入经典否定:

$$L \leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n.$$

  其中， L 和 L_i 是文字，即原子或原子的否定 (¬)。
- 对于含经典否定的正逻辑程序，仍然可以定义极小 Herbrand 模型，只不过此时 Herbrand 解释对应为文字的集合。
- 程序的 Herbrand 模型是一致的（不同时存在 p 和 $\neg p$）或者是文字的全体集合 Lit。在此约束下，逻辑程序保持前面介绍的各种性质。

经典否定

- 前面的介绍中，没有涉及经典否定 $\neg$。在程序中其实可以加入经典否定:
$$L \leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n.$$
其中， L 和 L_i 是文字，即原子或原子的否定 $(\neg)$。
- 对于含经典否定的正逻辑程序，仍然可以定义极小 Herbrand 模型，只不过此时 Herbrand 解释对应为文字的集合。
- 程序的 Herbrand 模型是一致的（不同时存在 p 和 $\neg p$）或者是文字的全体集合 Lit。在此约束下，逻辑程序保持前面介绍的各种性质。
- 程序的 Answer set 为一个文字集 S，满足 $S = AS(P^S)$。

Outline

1. 关于暑期讨论班
2. 逻辑程序
3. 稳定模型
4. 相关阅读材料

下一阶段阅读文献

Gelfond, M. and Lifschitz, V.
The Stable Model Semantics For Logic Programming.
In *Proceedings of the Fifth International Conference on Logic Programming (ICLP-88)*, pages 1070–1080, 1988.

Gelfond, M. and Lifschitz, V.
Classical Negation in Logic Programs and Disjunctive Databases.
New Generation Computing 9:365–385, 1991.