

# 一階述語論理

Chapter 8

# 概要

- なぜ一階述語論理か
- 一階述語論理の構文と意味
- 一階述語論理を使う
- 一階述語論理でのWumpusの世界
- 一階述語論理での知識工学

# 命題論理の利点と欠点

- ☺ 命題論理は**宣言的**である
- ☺ 命題論理は選言や否定を用いて部分的な情報を表現できる
  - (多くのデータ構造やデータベースではこのようなことはない)
- ☺ 命題論理は**合成的**である:
  - $B_{1,1} \wedge P_{1,2}$  の意味は  $B_{1,1}$  と  $P_{1,2}$  の意味から導出される
- ☺ 命題論理の意味は**内容に独立**である
  - (意味が内容に従属する自然言語とは異なる)
- ☹ 命題論理は非常に限定された表現力を有する
  - (自然言語とは異なる)
  - 例: “窪みは隣の四角にそよ風を起こす” ということは表現できない
    - 各四角に対して一つの文章を書けることを除いて

# 一階述語論理

- 命題論理は世界が**事実**を含んでいると仮定している
- 一階述語論理は(自然言語のように)世界が次のことを含んでいると仮定する
  - **もの**: people, houses, numbers, colors, baseball games, wars, ...
  - **関係**: red, round, prime, brother of, bigger than, part of, comes between, ...
  - **関数**: father of, best friend, one more than, plus, ...

# 一階述語論理の構文: 基本要素

- 定数 KingJohn, 2, UNI,...
- 述語 Brother, >,...
- 関数 Sqrt, LeftLegOf,...
- 変数 x, y, a, b,...
- 結合子  $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- 等号 =
- 限定子  $\forall, \exists$

# 原始文

原始文 = 述語(項<sub>1</sub>, ..., 項<sub>n</sub>) | 項<sub>1</sub> = 項<sub>2</sub>

項 = 関数(項<sub>1</sub>, ..., 項<sub>n</sub>) | 定数 | 変数

- 例: *Brother(KingJohn, RichardTheLionheart) > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

# 複文

- 複文は結合子を使って原始文から作られる

$$\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2,$$

例:  $Sibling(KingJohn, Richard) \Rightarrow$   
 $Sibling(Richard, KingJohn)$

$$>(1,2) \vee \leq (1,2)$$

$$>(1,2) \wedge \neg >(1,2)$$

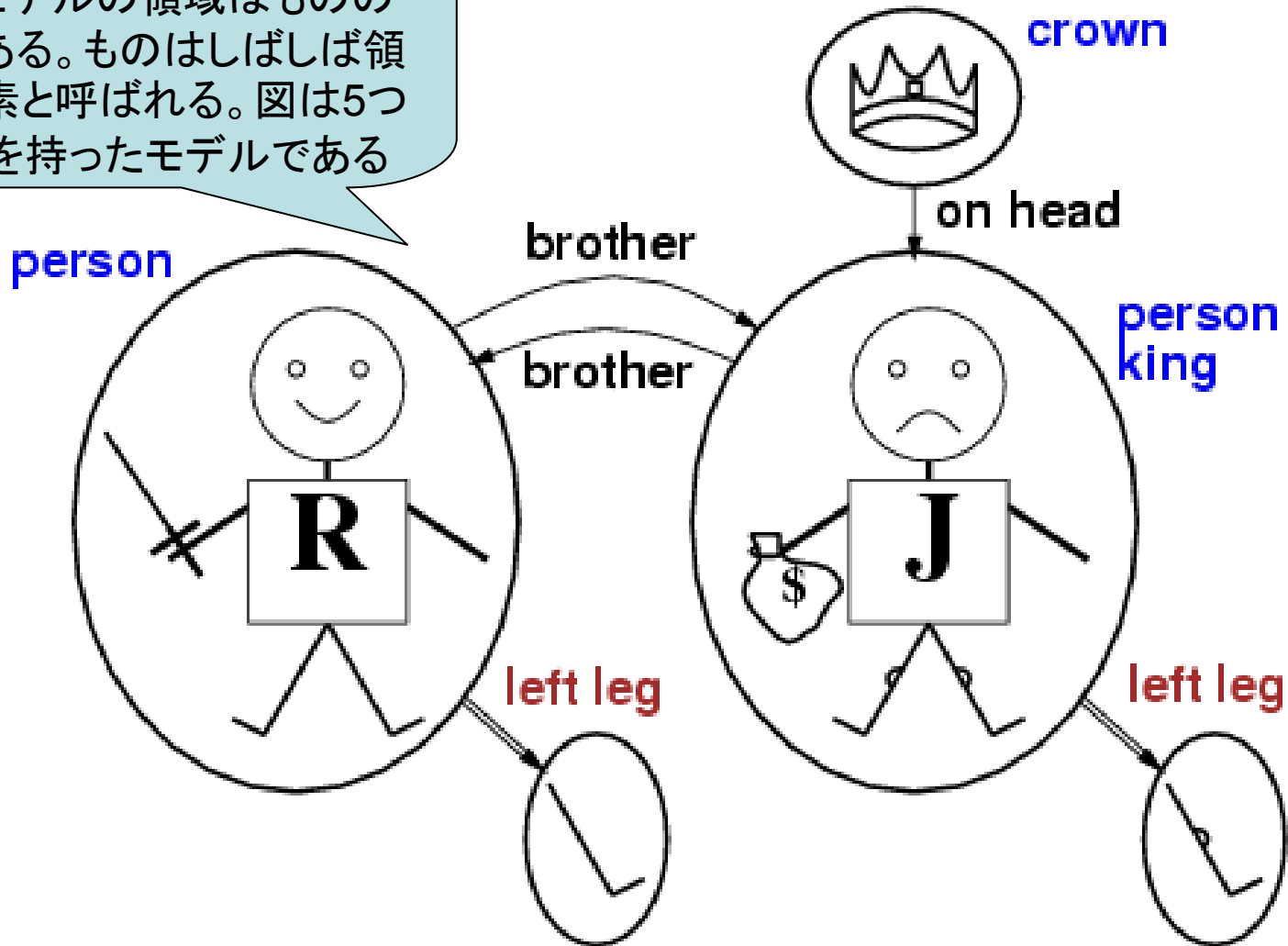
# 一階述語論理での真

- 文はモデルと解釈に関して真である
- モデルはもの(領域の要素)とそれらの関係を含んでいる
- 解釈は次への参照を明確にする
  - 定数のシンボル → もの
  - 述語のシンボル → 関係
  - 関数のシンボル → 関数関係
- 原始文  $predicate(term_1, \dots, term_n)$  は真であるときかつそのときに限り  $term_1, \dots, term_n$  によって参照されているものが  $predicate$  によって参照されている関係の中にある

# 一階述語論理のモデル

例:

モデルはもの(オブジェクト)を持ち、モデルの領域はものの集合である。ものはしばしば領域の要素と呼ばれる。図は5つのものを持ったモデルである



# 全称限定子

- $\forall$  <変数> <文>
- UNIの全ての人is smartである:  
 $\forall x \text{ At}(x, \text{UNI}) \Rightarrow \text{Smart}(x)$
- $\forall x P$ はモデル  $m$  で真であるときかつそのときに限りモデル内の可能なそれぞれのもの  $x$  に対して真である
- 簡単に言うと  $P$  の即値の連言に同値である
  - $\text{At}(\text{KingJohn}, \text{UNI}) \Rightarrow \text{Smart}(\text{KingJohn})$
  - $\wedge \text{At}(\text{Richard}, \text{UNI}) \Rightarrow \text{Smart}(\text{Richard})$
  - $\wedge \text{At}(\text{UNI}, \text{UNI}) \Rightarrow \text{Smart}(\text{UNI})$
  - $\wedge \dots$

# 犯しやすい誤り

- 典型的に $\Rightarrow$ は $\forall$ を伴っての主要な結合子である
- 良くある誤り:  $\forall$ を伴っての主要な結合子として $\wedge$ を用いる:

$$\forall x \text{ At}(x, \text{UNI}) \wedge \text{Smart}(x)$$

は次のことを意味する “すべての人がUNIにいるそして全ての人はsmartである”

# 存在限定子

- $\exists$  <変数> <文>
- UNIのある人はsmartである:  
 $\exists x \text{ At}(x, \text{UNI}) \wedge \text{Smart}(x)$
- $\exists x P$  はモデル  $m$  で真であるときかつそのときに限り  
モデル内のある可能なもの  $x$  に対して真である
- 簡単に言うと  $P$  の **即値の選言** に同値である
  - $\text{At}(\text{KingJohn}, \text{UNI}) \wedge \text{Smart}(\text{KingJohn})$
  - ✓  $\text{At}(\text{Richard}, \text{UNI}) \wedge \text{Smart}(\text{Richard})$
  - ✓  $\text{At}(\text{UNI}, \text{UNI}) \wedge \text{Smart}(\text{UNI})$
  - ✓ ...

# 犯しやすい誤り

- 典型的に $\wedge$ は $\exists$ を伴っての主要な結合子である
- 良くある誤り:  $\exists$ を伴っての主要な結合子として  $\Rightarrow$  を用いる:

$$\exists x \text{ At}(x, \text{UNI}) \Rightarrow \text{Smart}(x)$$

は“UNIにいない任意の人がいる”とき真である

# 限定子の性質

- $\forall x \forall y$  is the same as  $\forall y \forall x$
- $\exists x \exists y$  is the same as  $\exists y \exists x$
- $\exists x \forall y$  is **not** the same as  $\forall y \exists x$
- $\exists x \forall y \text{ Loves}(x,y)$ 
  - “There is a person who loves everyone in the world”
- $\forall y \exists x \text{ Loves}(x,y)$ 
  - “Everyone in the world is loved by at least one person”
- **Quantifier duality:** each can be expressed using the other
- $\forall x \text{ Likes}(x, \text{IceCream})$                        $\neg \exists x \neg \text{Likes}(x, \text{IceCream})$
- $\exists x \text{ Likes}(x, \text{Broccoli})$                        $\neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

# 等号

- $term_1 = term_2$  は与えられた解釈の中で真であるときかつそのときに限り  $term_1$  と  $term_2$  は同じものを参照する
- 例: definition of *Sibling* in terms of *Parent*:  
$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \neg (m = f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$$

# 一階述語論理の使用

The kinship domain:

- Brothers are siblings

$$\forall x,y \text{ Brother}(x,y) \Leftrightarrow \text{Sibling}(x,y)$$

- One's mother is one's female parent

$$\forall m,c \text{ Mother}(c) = m \Leftrightarrow (\text{Female}(m) \wedge \text{Parent}(m,c))$$

- “Sibling” is symmetric

$$\forall x,y \text{ Sibling}(x,y) \Leftrightarrow \text{Sibling}(y,x)$$

# 一階述語論理の使用

要素なし

要素あり: 集合 $s_2$ と $s$ の論理積が $s_2$ の要素 $x$ となる $s_2$ と $x$ が存在する。

The set domain:

- $\forall s \text{ Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \{x|s_2\})$
- $\neg \exists x, s \{x|s\} = \{\}$
- $\forall x, s \ x \in s \Leftrightarrow s = \{x|s\}$
- $\forall x, s \ x \in s \Leftrightarrow [\exists y, s_2 (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$
- $\forall s_1, s_2 \ s_1 \subseteq s_2 \Leftrightarrow (\forall x \ x \in s_1 \Rightarrow x \in s_2)$
- $\forall s_1, s_2 \ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- $\forall x, s_1, s_2 \ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- $\forall x, s_1, s_2 \ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

# 一階述語論理の使用(集合)

- 集合は空集合と集合に何かをくっつけて作られた集合である:

$$\forall s \text{ Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \{x|s_2\})$$

- 空集合はそれにくっつけられた要素を持たない:

$$\neg \exists x, s \{x|s\} = \{\}$$

- 集合にある要素をくっつけてもそれは変わらない:

$$\forall x, s \ x \in s \Leftrightarrow s = \{x|s\}$$

- 集合の要素はそれにくっつけられた要素のみで成り立つ:

$$\forall x, s \ x \in s \Leftrightarrow [\exists y, s_2 (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$$

- 集合はある集合の部分集合であるときかつそのときに限り最初の集合の要素はすべて次の集合の要素である:

$$\forall s_1, s_2 \ s_1 \subseteq s_2 \Leftrightarrow (\forall x \ x \in s_1 \Rightarrow x \in s_2)$$

- 二つの集合が等価であるときかつそのときに限りお互いに他の集合の部分集合である:

$$\forall s_1, s_2 \ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$$

- 二つの集合の共通のものであるときかつそのときに限り両方の集合の要素である:

$$\forall x, s_1, s_2 \ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$$

- 二つの集合の和であるときかつそのときに限り少なくともどちらかの集合の要素である:

$$\forall x, s_1, s_2 \ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$$

# 一階述語論理の知識ベースを用いて の相互作用

- wumpus-worldのエージェントが一階述語論理の知識ベースKBを使い $t=5$ で(輝いたものではなく)悪臭とそよ風を知覚したとする：  
    Tell(KB, Percept([Smell, Breeze, None], 5))  
    Ask(KB,  $\exists a$  BestAction(a, 5))
- 即ち、知識ベースは $t=5$ での最善の動作を伴意するか？
- 答え: Yes, {a/Shoot} ← 置換 (binding list)
- 文 $S$ と置換 $\sigma$ が与えられて
- $S\sigma$  は $S$ に $\sigma$ をプラグインした結果を示す  
     $S = \text{Smarter}(x, y)$   
     $\sigma = \{x/\text{Hillary}, y/\text{Bill}\}$   
     $S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill})$
- Ask(KB,  $S$ ) は $KB \vdash \sigma$ であるような $\sigma$ を返す

# wumpusの世界での知識ベース

- 知覚

- $\forall t, s, b \text{ Percept}([s, b, \text{Glitter}], t) \Rightarrow \text{Glitter}(t)$

- 反射運動

- $\forall t \text{ Glitter}(t) \Rightarrow \text{BestAction}(\text{Grab}, t)$

# 隠れた性質を引き出す

- $\forall x,y,a,b \text{ Adjacent}([x,y],[a,b]) \Leftrightarrow$   
 $[a,b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

四角の性質:

- $\forall s,t \text{ At}(\text{Agent},s,t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$

四角は窪みの近くではそよ風がある:

– 診断的ルール---影響から原因を推察

$$\forall s \text{ Breezy}(s) \Rightarrow \exists r \text{ Adjacent}(r,s) \wedge \text{Pit}(r)$$

– 因果的ルール---原因から影響を推察

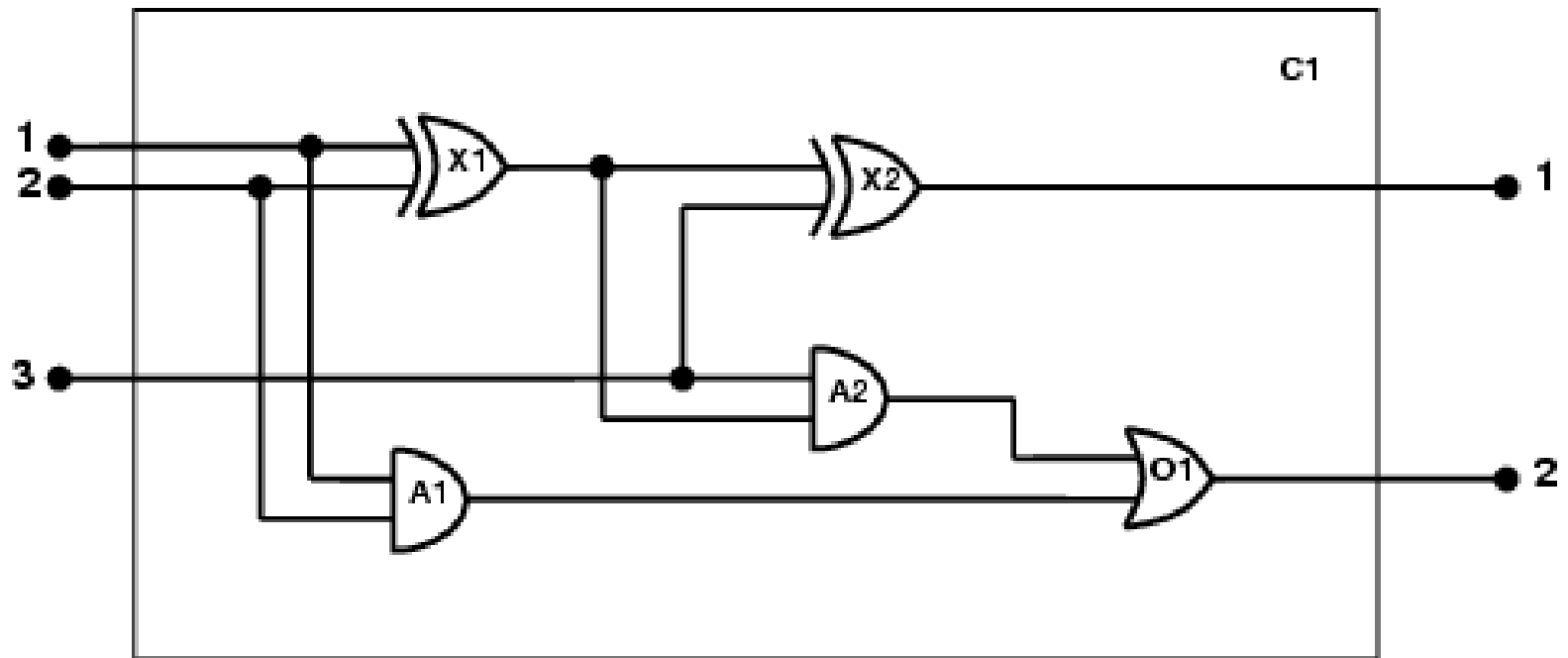
$$\forall r \text{ Pit}(r) \Rightarrow [\forall s \text{ Adjacent}(r,s) \Rightarrow \text{Breezy}(s)]$$

# 一階述語論理での知識工学

1. 仕事を同定する
2. 関連する知識を集める
3. 述語、関数、定数の語彙を決定する
4. 領域についての一般的な知識をコード化する
5. 特定の問題のインスタンスに対する記述をエンコードする
6. インターフェースの手続きに質問を出し答えを得る
7. 知識ベースをデバッグする

# 電子回路の領域

One-bit full adder



# 電子回路の領域

1. 仕事を同定する
  - 回路は実際に正しく加算をするか (circuit verification)
2. 関連する知識を集める
  - 線とゲートで構成; ゲートの種類 (AND, OR, XOR, NOT)
  - 無関係: size, shape, color, cost of gates
3. 述語、関数、定数の語彙を決定する
  - Alternatives:  
Type( $X_1$ ) = XOR  
Type( $X_1$ , XOR)  
XOR( $X_1$ )

# 電子回路の領域

## 4. 領域についての一般的な知識をコード化する

- $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
- $\forall t \text{ Signal}(t) = 1 \vee \text{Signal}(t) = 0$
- $1 \neq 0$
- $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$
- $\forall g \text{ Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 1 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n, g)) = 1$
- $\forall g \text{ Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 0 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n, g)) = 0$
- $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1, g)) \neq \text{Signal}(\text{In}(2, g))$
- $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1, g)) \neq \text{Signal}(\text{In}(1, g))$

# 電子回路の領域

## 5. 特定の問題のインスタンスに対する記述をエンコードする

Type( $X_1$ ) = XOR

Type( $X_2$ ) = XOR

Type( $A_1$ ) = AND

Type( $A_2$ ) = AND

Type( $O_1$ ) = OR

Connected(Out(1, $X_1$ ),In(1, $X_2$ ))

Connected(In(1, $C_1$ ),In(1, $X_1$ ))

Connected(Out(1, $X_1$ ),In(2, $A_2$ ))

Connected(In(1, $C_1$ ),In(1, $A_1$ ))

Connected(Out(1, $A_2$ ),In(1, $O_1$ ))

Connected(In(2, $C_1$ ),In(2, $X_1$ ))

Connected(Out(1, $A_1$ ),In(2, $O_1$ ))

Connected(In(2, $C_1$ ),In(2, $A_1$ ))

Connected(Out(1, $X_2$ ),Out(1, $C_1$ ))

Connected(In(3, $C_1$ ),In(2, $X_2$ ))

Connected(Out(1, $O_1$ ),Out(2, $C_1$ ))

Connected(In(3, $C_1$ ),In(1, $A_2$ ))

# 電子回路の領域

6. インターフェースの手続きに質問を出し答えを得る

加算器の全ての終端での可能な値の集合は何か

$$\exists i_1, i_2, i_3, o_1, o_2 \text{ Signal(In}(1, C_1)) = i_1 \wedge \text{Signal(In}(2, C_1)) = i_2 \\ \wedge \text{Signal(In}(3, C_1)) = i_3 \wedge \text{Signal(Out}(1, C_1)) = o_1 \wedge \\ \text{Signal(Out}(2, C_1)) = o_2$$

7. 知識ベースをデバッグする

1 ≠ 0のような表明は削ることができるか

# まとめ

- 一階述語論理:
  - ものとの関係は意味での根源である
  - 構文: 定数、関数、述語、等号、限定子
- 増大した表現力: wumpusの世界を定義するのに十分