人工知能入門
第3回

藤田 悟

黄　潤和

前回の復習: 知識の種類

- **is-a 型**: 上位概念と下位概念を結ぶ
  - ニワトリは鳥である
- **has-a 型**: 属性・部分として保有する
  - ニワトリは羽がある
- 性質を述べる
  - 鳥は空を飛ぶ
- 特定知識
  - あのニワトリの名前は「コッコちゃん」である

前回の復習: 知識の論理式表現

- 知識を論理式(命題論理)の体系として表現する
  - P: ニワトリは鳥である
  - Q: 鳥は空を飛ぶ
  - R: ニワトリは空を飛ぶ
  - $P \wedge Q$: ニワトリは鳥である かつ 鳥は空を飛ぶ
  - $P \vee Q$: ニワトリは鳥である または 鳥は空を飛ぶ
  - $\neg P$: ニワトリは鳥でない
  - $P \wedge Q \Rightarrow R$
    - ニワトリは鳥である かつ 鳥は空を飛ぶ ならば、ニワトリは空を飛ぶ

今回学ぶこと

- 述語論理による知識表現
  - 論理式に変数を導入し、表現力を高める
  - 限量子を導入する
- 述語論理による推論
  - 述語論理を用いて、論理的に正しい推論を行う

述語論理による知識表現

命題論理の限界

- 命題論理では、全ての知識を論理式の項として記述しなければならなかった
  1. P: ニワトリは鳥である
  2. Q: ニワトリは空を飛ぶ
  3. P⇒Q: ニワトリが鳥であるならば、ニワトリは空を飛ぶ
- カモについて知識を加える場合、カモについて、全ての知識が必要になり、知識量が膨大になる。
  1. P: カモは鳥である
  2. Q: カモは空を飛ぶ
  3. P⇒Q: カモが鳥であるならば、カモは空を飛ぶ

述語論理

- 論理式の要素を**述語とパラメータで表現**できるように拡張したものを、述語論理と呼ぶ。
  - 鳥(ニワトリ): ニワトリは鳥である
  - 飛ぶ(ニワトリ): ニワトリは空を飛ぶ
  - 鳥(ニワトリ)⇒飛ぶ(ニワトリ): ニワトリが鳥であれば、ニワトリは空を飛ぶ
  - 鳥(カモ): カモは鳥である
- パラメータは変数にしてもよい
  - 鳥(x)⇒飛ぶ(x): xが鳥ならば、xは空を飛ぶ
    - 変数xを用いれば、ニワトリとカモの両者について知識を書く必要がなくなる

述語論理による「ニワトリは飛べるか」

- 鳥(x) ⇒ 飛ぶ(x)
- 鳥(ニワトリ)
- 鳥(カモ)

- 変数xにニワトリを代入すると、飛ぶ(ニワトリ)が成立し、変数xにカモを代入すると、飛ぶ(カモ)が成立する。

複数のパラメータを持つ述語論理式

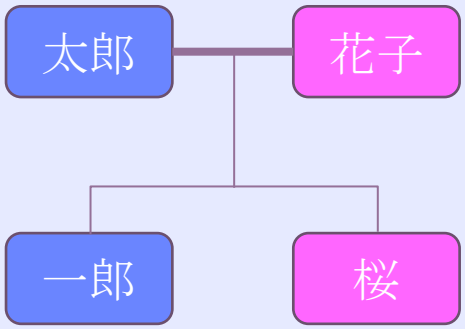
- 父(太郎, 一郎): 太郎は一郎の父である
- 父(太郎, 二郎): 太郎は二郎の父である
- 父(太郎, x): 父が太郎であるx
  - 変数 x は、一郎、二郎の二つの可能性がある
- 父(武蔵, 太郎): 武蔵は太郎の父である
- 祖父(x, z) ⇒父(x, y) ∧ 父(y, z)
  - xがzの祖父ならば、xがyの父で、yがzの父である
  - 武蔵、太郎、一郎の関係から、祖父(武蔵, 一郎)を導くことができる。

限量子

- 述語論理は、P(x)⇒Q(x) のように表現するが、これは、全てのxについて述べた知識なのか、それとも、この論理式が成り立つある特定の x が存在するという意味なのだろうか？
- 対象が全てなのか、存在するのかを明確に記述するのが限量子である
  - ∀x: P(x)　全てのxについてP(x)が成り立つ(全称限量子)
  - ∃x: P(x)　P(x)が成立するxが少なくとも一つ存在する(存在限量子)
  - 特に記述がなければ、全称限量子を前提とする

述語論理を用いた推論

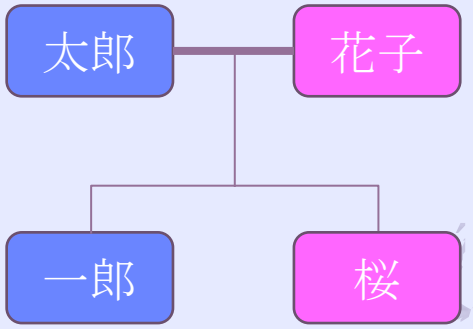
- 親(太郎, 一郎)
- 親(太郎, 桜)
- 親(花子, 一郎)
- 親(花子, 桜)
- 男(太郎)
- 男(一郎)
- 女(花子)
- 女(桜)
- 父(x, y) ⇒ 親(x, y) ∧ 男(x)
- 娘(y, x) ⇒ 親(x, y) ∧ 女(y)



- 一郎の父は? 父(x, 一郎)
- 太郎の娘は? 娘(x, 太郎)

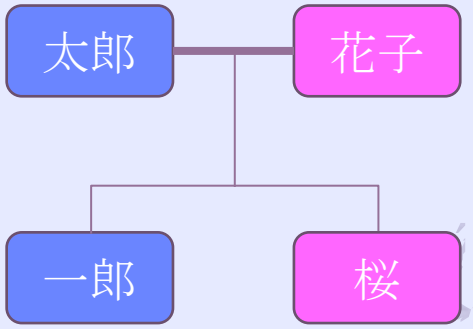
推論の過程1

- 父(x, 一郎)
  - 適用: 父(x, y) ⇒ 親(x, y) ∧ 男(x)
  - 父(x, 一郎) ⇒ 親(x, 一郎) ∧ 男(x)
  - 親(x, 一郎)∧男(x)
  - 親(x, 一郎)|x∈{太郎,花子}∧男(x)
  - 親(太郎, 一郎)∧男(太郎)
- 父(太郎, 一郎) … x = 太郎



推論の過程2

- 娘(x, 太郎)
  - 適用: 娘(y, x) ⇒ 親(x, y) ∧ 女(y)
  - 娘(y, 太郎) ⇒親(太郎, y) ∧ 女(y)
  - 親(太郎, x) ∧ 女(x)
  - 親(太郎, x) | x∈{一郎, 桜}∧女(x)
  - 親(太郎, 桜)∧女(桜)
- 娘(桜, 太郎) … x = 桜



(演習1) 述語論理

- 次の定義を述語論理で記述せよ。ただし、例題に出てきた述語(親, 男, 女)を利用してよい
  - 息子(x, y)
  - 母(x, y)
- 定義した述語を使って、次の結果を導け。推論の過程も例題同様に示せ。
  - 息子(一郎, y)
  - 母(x, 一郎)

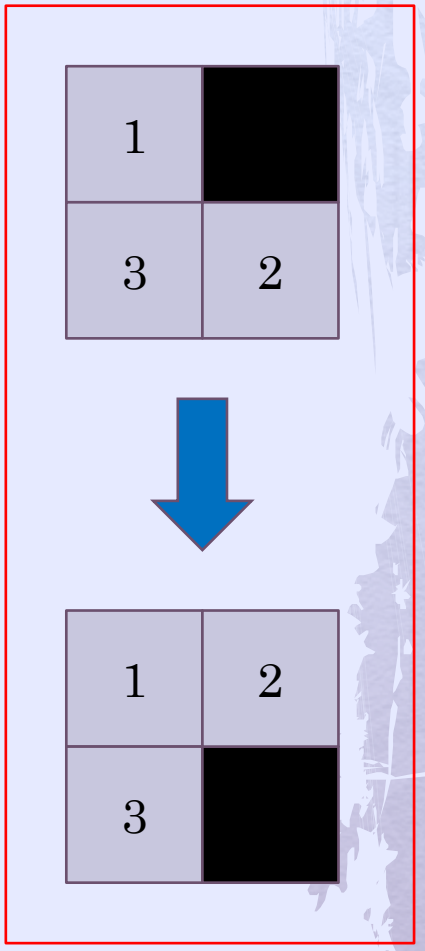
ゲームの論理表現

- 3パズル(15パズルの簡易形)
- 状態と動作を論理式で表現する

  p(ターン数, タイル番号, x座標, y座標)

  - p(1,1,1,1)
  - p(1,2,2,2)
  - p(1,3,1,2)
  - p(1,0,2,1)
  - p(i+1,n,2,y)∧p(i+1,0,1,y)
    ⇒p(i,n,1,y)∧p(i, 0,2,y)∧right(i)
  - p(i+1,n,1,y)∧p(i+1,0,2,y)
    ⇒p(i,n,2,y)∧p(i, 0,1,y)∧left(i)
  - p(i+1,n,x,1)∧p(i+1,0,x,2)
    ⇒p(i,n,x,2)∧p(i, 0,x,1)∧up(i)
  - p(i+1,n,x,2)∧p(i+1,0,x,1)
    ⇒p(i,n,x,1)∧p(i, 0,x,2)∧down (i)

(1,1)	(2,1)
(1,2)	(2,2)

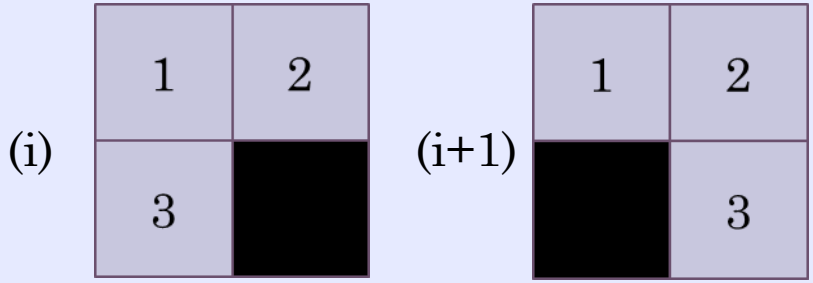


詳細な説明

- p(ターン数, タイル番号, x座標, y座標)
  - p(1,1,1,1) が真とは、1ターン目に、番号1番のタイルが、(x,y) = (1,1)の座標に置かれている状態を表す。
  - 番号0番のタイルは、空き空間であることを示す。
  - ターンごとに、タイルが置いてある場所は異なるので全てのターンiに対して、タイルの番号nごとに、p(i,n,x,y)を真とする(x,y)の組合せが一つ存在する。

詳細な説明

- タイルを右に動かす動作を right(i)とする
  - p(i+1,n,2,y)∧p(i+1,0,1,y)
    ⇒p(i,n,1,y)∧p(i,0,2,y)∧right(i)
  - 前提条件として、空きタイルが、右(2,y)にあり、その左側のタイル(1,y)と場所を交換する。



  - 正確には、動きと関係のないタイルは、次のターン(i+1)でも同じ位置にいることを表す真偽値も必要であるが、ここでは省略する。

ゲームの推論

- <<ゴール>>
  p(i,1,1,1)∧p(i,2,2,1)∧p(i,3,1,2)∧p(i,0,2,2)
- ⇒p(i-1,1,1,1)∧p(i-1,2,2,2)∧
  p(i-1,3,1,2)∧p(i-1,0,2,1)∧up(i-1)
- ⇒p(1,1,1,1)∧p(1,2,2,1)∧
  p(1,3,1,2)∧p(1,0,2,2)∧up(1)
- ⇒<<初期状態>>∧up(1)
- したがって、up(1)を実行することで、
  第2ターンで、ゴールに到達できる

論理式表現の長所・短所

- 長所
  - 論理式で表現できれば、正しく推論することで解に導くことができる。
  - 宣言的な知識(論理式)だけで、問題解決を行える。
  - 推論するための手続き的知識が不要。
- 短所
  - 論理式が直観的でない場合がある。
  - 推論の高速化など、手続的な最適化が困難。

PROLOG

Prolog

- 述語論理に従って、宣言的な知識を記述することによって、計算を行うプログラミング言語
  - 「第五世代コンピュータ」という日本の国家プロジェクトでは、主のプログラミング言語として、Prolog の動作する並列コンピュータが開発された。
- 下記のアドレスで、簡単なプログラムを実行可能
  - http://ioctl.org/logic/prolog-latest

Prolog の構文

- 真実を記述する時には、単独で要素を書き、period "." で終わる行を書く。
  - parent("Taro", "Ichiro").
- 文字列は、" で囲む。
- 変数は大文字で始める。
- 推論規則を、述語論理として書く。
  - 「ならば」は、":-"　と書く。(正しくは、「ならば」と方向が逆であり、左辺が結論のつもりで利用する)
  - 論理積の条件は、comma "," でつなぐ。
  - 最後に period "." で終わる
  - father(X, Y) :- parent(X, Y), man(X).

Prolog プログラム例

```
parent("Taro", "Ichiro").
parent("Taro", "Jiro").
parent("Taro", "Sakura").
parent("Hanako", "Ichiro").
parent("Hanako", "Jiro").
parent("Hanako", "Sakura").
man("Taro").
man("Jiro").
man("Ichiro").
woman("Hanako").
woman("Sakura").
father(X,Y) :- parent(X,Y), man(X).
mother(X,Y) :- parent(X,Y), woman(X).
daughter(X,Y) :- parent(Y,X), woman(X).
son(X,Y) :- parent(Y,X), man(X).
```

実行結果

Query
son(X,"Taro").
X = "Ichiro"
X = "Jiro"

Query
mother(X, "Sakura").
X = "Hanako"

Query
father("Taro", "Jiro").
true

(演習2)

- 本資料にある prolog のサンプルコードを入力し、次の動作を確認し、出力を報告せよ。
  - parent(X, "Ichiro").
  - parent("Taro", X).

(演習3)

- 演習1, 教科書 p36,に回答せよ

息子(一郎, y)
母(x, 一郎)

son("Ichiro", Y).
mother(X, "Ichiro")

課題1

小さな prolog プログラムを作成し、動作を確認せよ。例えば、

(1) 次の程度のプログラムで十分である。
他には、祖父・祖母の推論、兄弟などの推論でも良い。

(2)

```
bird("Niwatori").
fly(X) :- bird(X).  (順番に注意せよ)
```