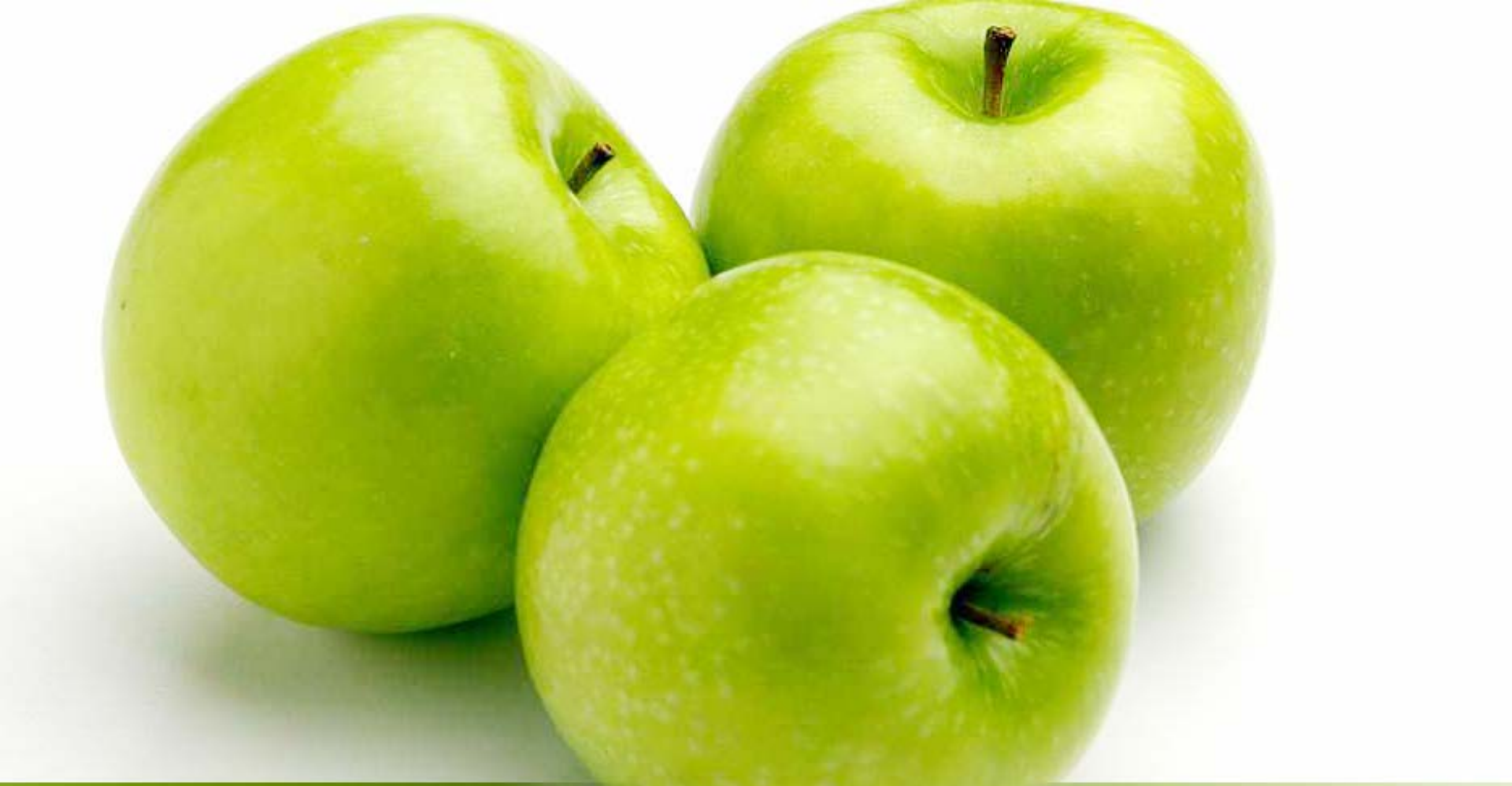

知識工学　第7回

二宮　崇

教科書と資料

- 教科書
  - Artificial Intelligence: A Modern Approach (3rd Edition): Stuart Russell, Peter Norvig (著), Prentice Hall, 2009

- この講義のウェブサイト

http://aiweb.cs.ehime-u.ac.jp/~ninomiya/ke/

本日の講義内容

- 一階述語論理による推論(§9)
  - 命題論理 対 一階述語論理 (§9.1)
    - 限量子に対する推論規則(§9.1.1)
    - 命題論理への帰着(§9.1.2)
  - 単一化と持ち上げ(§9.2)
    - 一階推論規則(§9.2.1)
    - 単一化(§9.2.2)

一階述語論理による推論(§9)
命題論理 対 一階述語論理 (§9.1)

- 一階述語論理においてはモデルが無数に存在するためモデル検査による論理的同値関係の判定を行うことはできない。
- モデルを有限に限ったとしても、その解釈は爆発的数の解釈が存在することになる。

命題論理 対 一階述語論理 (§9.1)

- 命題論理と一階述語論理における論理的同値関係と伴意関係

	命題論理	一階述語論理
論理的同値関係 $\alpha \equiv \beta$	真理値表が同一になる	
	$\alpha \Leftrightarrow \beta$がトートロジー	$\alpha \Leftrightarrow \beta$がトートロジー
	$\alpha \models \beta$かつ$\beta \models \alpha$	$\alpha \models \beta$かつ$\beta \models \alpha$
伴意関係 $\alpha \models \beta$	$\alpha \equiv \alpha \wedge \beta$	$\alpha \equiv \alpha \wedge \beta$
	$\alpha \Rightarrow \beta$がトートロジー	$\alpha \Rightarrow \beta$がトートロジー

命題論理 対 一階述語論理 (§9.1)

- モデル検査を用いることができなくても、その伴意関係の公理を与えることは出来そう。
  - αが真であるとき、βも常に真であるならば、伴意関係が成り立つ
  - 命題論理において真理値表を用いずとも推論ができたように、一階述語論理においても論理的同値関係、伴意関係の公理を定義すればそれらを用いることにより推論が可能。

限量子に対する推論規則(§9.1.1)

- 基礎項 (ground term)
  - 変数を含まない項。

- 代入 $SUBST(\theta, \alpha)$
  - 文αに代入θを適用する。
  - θは$\{v_1/g_1, \cdots, v_n/g_n\}$という形をしており、変数$v_i$に基礎項$g_i$を代入することを意味する。

限量子に対する推論規則(§9.1.1)

- 全称具体化 (universal instantiation, UI)
  - $\forall v\ \alpha$のvを任意の基礎項で置き換えた文αを伴意(推論)

$$\frac{\forall v\ \alpha}{SUBST(\{v/g\}, \alpha)}$$

例: $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$に対して、

- $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
- $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
- $King(Father(john)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$
- …

任意のvに対して文αが成り立つわけだから、vに任意の項を代入しても良い、ということ

限量子に対する推論規則(§9.1.1)

- 存在具体化 (existential instantiation, EI)
  - $\exists v\ \alpha$のvを新しい定数kで置き換えた文αを伴意(推論)。

$$\frac{\exists v\ \alpha}{SUBST(\{v/k\}, \alpha)}$$

例: $\exists x\ Crown(x) \wedge OnHead(x, John)$に対し、次の文を伴意

- $Crown(C) \wedge OnHead(C, John)$

ただし、Cは知識ベースのどこにも現れていない新しい定数でなければならない。この新しい定数はスコーレム定数 (Skolem constant) と呼ばれる。

あるvに対して、文αが成り立つわけだから、そのあるvに定数Cという名前をつけてやった、ということ。

命題論理への帰着(§9.1.2)

- 全称具体化と存在具体化を用いて、限量子のついた文から限量子のついていない文を推論
- 限量子のついていない新しい文を知識ベースに追加(伴意関係)
- 限量子のついていない文だけを用いて推論することにより、一階述語論理を命題論理に帰着

  →命題論理の推論問題として一階述語論理の問題を解く

命題論理への帰着(§9.1.2)

- 限量子のつかない原子文は変数を持たない原子基礎文 (ground atomic sentence)
- 異なる原子基礎文 (ground atomic sentence) には異なる真偽値を自由に割り当てられる
  → 原子基礎文=命題記号と考える
- よって、一階述語論理の推論＝命題論理の推論

命題論理への帰着(§9.1.2)

- 例: 次の知識ベースが成り立つとき、$Evil(John)$が成り立つかどうか？

R_1: $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

R_2: $King(John)$

R_3: $Greedy(John)$

R_4: $Brother(Richard, John)$

このとき、UIより、

R_5: $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

が得られ、R_2とR_3とR_5に対し、モーダスポーネンスを適用

R_6: $Evil(John)$

命題論理への帰着(§9.1.2)

- この手法は命題論理化 (propositionalization) と呼ばれる。
- いかなる一階述語論理の知識ベースおよび質問は命題論理に置き換えることができるような気がするかもしれないが、実はそれは難しい。
  - 問題は、関数を含む項は無限に入れ子になった項(例えば、$Father(Father(\ldots\ (Father(John))\ \ldots))$)
  - UIによって、無限に新しい知識を生成することができてしまう。

命題論理への帰着(§9.1.2)

- しかし、伴意関係にある知識ベースと質問に対しては有限の大きさの命題論理に必ず置き換えることができることがわかっている。(エルブランの定理)
  - →有限時間でその伴意関係が成り立つかどうか判定できる
- 一階述語論理に対する伴意関係の問題は、半決定的(semidecidable)
  - チューリング機械の停止問題と同じ
  - 伴意関係にある文に対してはyesと答えるアルゴリズムが存在する
  - 伴意関係にない文に対してnoと答えるアルゴリズムは存在しない。

単一化と持ち上げ(§9.2)

- 命題論理化による推論では、全称具体化と存在具体化をどのように用いれば命題化できるのか、命題化するところまでが難しい。
- 先ほどの例は答えまですぐにたどり着く推論を行ったが、例えば、次のような推論をしたとしたらどうだろう？

$$King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$$

  - 何か害をなすわけではないが、無駄な推論

➡ 一階述語論理の形のままで推論する規則をいくつか用意することで無駄な推論をしなくてすむように工夫しよう。

持ち上げ(lifting)

- 命題論理の推論規則を一階述語論理の推論規則にすることを「持ち上げ」という

- 持ち上げにより得られる推論規則を用いて、直接、一階述語論理の推論をすることを考える

一階推論規則(§9.2.1)

- 一般化モーダスポーネンス
  - ある原子文p_i, p'_iとqに対して、全てのiに対して$SUBST(\theta, p'_i) = SUBST(\theta, p_i)$となる代入が存在するとき、次の推論が成り立つ。

$$\frac{p'_1, p'_2, \cdots, p'_n, (p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)}{SUBST(\theta, q)}$$

一階推論規則(§9.2.1)

- 例

R_1: $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

R_2: $King(John)$

R_3: $\forall y\ Greedy(y)$

この知識ベースに対し、

$$\frac{King(John), Greedy(y), King(x) \wedge Greedy(x) \Rightarrow Evil(x)}{SUBST(\{x/John, y/John\}, Evil(x)) = Evil(John)}$$

となり、$Evil(John)$を直接推論することができる。命題論理化に代わるこの推論規則はモーダスポーネンスの持ち上げ(lifting) 版と呼ばれる。

単一化(§9.2.2)

- 単一化
  - 異なる論理表現を同一にする代入を見つけること
    - つまり、持ち上げ推論規則における代入θを自動的に計算すること
  - 単一化のアルゴリズムには様々な手法があるが、グラフ単一化の手法が一般的。
  - いずれにせよ、単一化のアルゴリズム$UNIFY$は二つの文p, qを受け取り$SUBST(\theta, p) = SUBST(\theta, q)$となる代入$\theta$を返す(または$SUBST(\theta, p)$を返す)。つまり、

$$UNIFY(p, q) = \theta \; where \; SUBST(\theta, p) = SUBST(\theta, q)$$

単一化(§9.2.2)

- 例

R_1: $Knows(John, Jane)$

R_2: $\forall y\ Knows(y, Bill)$

R_3: $\forall y\ Knows(y, Mother(y))$

R_4: $\forall x\ Knows(x, Elizabeth)$

このとき、質問$\exists x\ Knows(John, x)$に対し、次を実行

$UNIFY\big(Knows(John, x), Knows(John, Jane)\big) = \{x/Jane\}$

$UNIFY\big(Knows(John, x), Knows(y, Bill)\big) = \{x/Bill, y/John\}$

$UNIFY\left(Knows(John, x), Knows\big(y, Mother(y)\big)\right) = \{y/John, x/Mother(John)\}$

$UNIFY\big(Knows(John, x), Knows(x, Elizabeth)\big) = fail?$

単一化(§9.2.2)

- 変数名の書き換え
  - $UNIFY\big(Knows(John, x), Knows(x, Elizabeth)\big)$が失敗?
    - 変数xに$John$と$Elizabeth$の両方を代入しているため。
  - 元々の式 $\forall x\ Knows(x, Elizabeth) \equiv \forall z\ Knows(z, Elizabeth)$ なので$\forall z\ Knows(z, Elizabeth)$に置き換える

$$UNIFY\big(Knows(John, x), Knows(z, Elizabeth)\big) = \{x/Elizabeth, z/John\}$$

  - このように変数名を書き換えておくことで、変数名の衝突問題を避けることができる

単一化(§9.2.2)

- 単一化のもう一つの問題
  - $UNIFY\big(Knows(John, x), Knows(y, z)\big)$に対する答えはどうなるであろう？

- 様々な可能な代入がありえる。例えば、次の二つがあり得る。
  - $\{y/John, x/z\}$ (つまり、$Knows(John, x)$)
  - $\{y/John, x/John, z/John\}$ (つまり、$Knows(John, John)$)

- この場合どちらの代入が好ましいのだろうか？

単一化(§9.2.2)

- 単一化のもう一つの問題
  - $UNIFY\big(Knows(John, x), Knows(y, z)\big)$に対する答えはどうなるであろう？
  - $\{y/John, x/z\}$ (つまり、$Knows(John, x)$)
    - より一般的でこの結果をまた別の推論に使える、という意味でも好ましい。
  - $\{y/John, x/John, z/John\}$ (つまり、$Knows(John, John)$)
    - 余計な推論(xとzを$John$としてしまうこと)が入っており、この代入が目的の推論に使えない場合はまったく役に立たない。

単一化(§9.2.2)

- 最汎単一化子 (most general unifier, MGU)
  - 項に対して代入を行うことで、項はより特殊な項となる。(逆に特殊になる前の項は一般な項という)
  - 二つの項に対する可能な代入のうち最も一般な項を返す代入のことを最汎単一化子 (MGU) と呼ぶ。
  - 単一化アルゴリズムは最汎単一化子を返す
  - 項の特殊-一般の関係は半順序関係となっており、単一化は上限(least upper bound)を求めていることになる。単一化において上限は唯一に定まる。

※上限は上界(upper bound)のうち最も下側の要素。情報数学Iの半順序関係を参照せよ。

単一化(§9.2.2)

- 参考: 単一化アルゴリズム

```
function UNIFY(x, y, θ):
  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if LIST?(x) and LIST?(y) then
    return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
  else return failure
```

単一化(§9.2.2)

- 参考: 単一化アルゴリズム (続き)

function UNIFY-VAR(var, x, θ):

if {var/val}$\in\theta$ then return UNIFY(val, x, θ)

else if {x/val}$\in\theta$ then return UNIFY(var, val, θ)

else if OCCUR-CHECK?(var, x) then return failure

else return add {var/x} to θ

※OCCUR-CHECK?(var,x)は変数varが項xの中に存在するかどうかチェックする述語である。これは単一化後、項の中にループ構造ができるどうかチェックしていることに相当する。ループができる場合は単一化に失敗する。多くの単一化アルゴリズムではoccur checkは省略されている。