2007年度 計算知能論A
推論と定理証明

上田 和紀
早稲田大学理工学部CS学科

21. 推論と定理証明

- Prologの動作にはどのような論理学的な背景があるのだろうか？
- Prologの動作原理を一般化して，一般の一階述語論理における証明ができないか？
  ➔ できる ☺.

21. 推論と定理証明

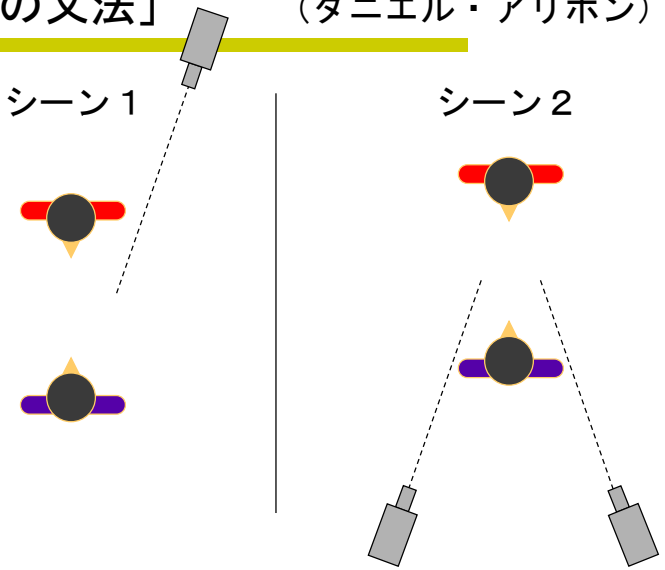
- 主な目標
  1. 表現とは
  2. 演繹的推論と非演繹的推論
  3. 閉論理式を用いた知識表現
  4. 論理的帰結の証明から充足不可能性の証明へ
  5. 一般の一階述語論理式を節形式に直す
  6. 複数の節を組合せて矛盾を導く：融合原理

21.1 表現 (representation) とは

側面 1 ：構文 (syntax)

- 記号表現の場合：
  - 字句（語彙）レベル — 記号
  - 構造レベル — 記号の配列
- 記号以外の表現にも構文（文法）はある！
  - 例： 表，グラフ
    アニメーション，映画 (film grammar)
    楽曲（例：ソナタ形式）

「映画の文法」　（ダニエル・アリホン）



21.1 表現 (representation)

側面２：操作 (operations)

- 普遍的操作
  - 表現の作成，格納，読出し，変更
- 個々の表現特有の操作
  - 例：　数式　→ 評価
    論理式の集合　→ 推論
    プログラム　→ 実行

側面３：意味 (semantics) ＝ 表現されたものの内容

21.1 知識表現言語の要件

- cf. 自然言語
  - 何でも表現できる
  - 高いモジュラリティ
  - 人間どうしのコミュニケーションの道具として発生
- cf.（手続き型）プログラム言語
  - 厳密（無曖昧，文脈独立）
  - 機械で扱える
- 両者の長所を兼ねたものが必要

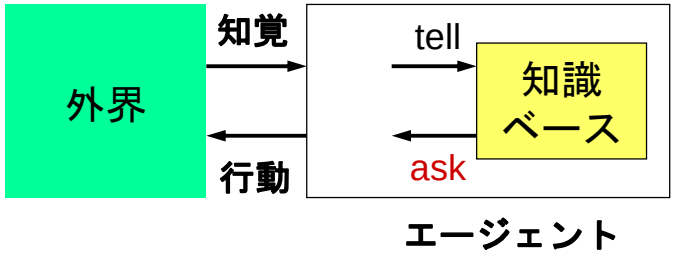
21.1 述語論理に基づく知識表現

- 汎用で表現力に富む
  - n (>2) 個のものの間の関係
  - 部分情報や一般的情報の表現（$\vee$, $\neg$, $\forall$, $\exists$）
- 推論操作の枠組が明確（演繹）
- 強固な数学的基礎

- 知識の構造化に言及していない
- 効率に言及していない
  - 表現の効率，認知／推論の効率

21.1 知的エージェント

- 知的エージェントは知識ベースをもつ
- 知識ベースへの操作（cf. メモリ読み書き）
  - tell:　知識の格納
  - ask:　推論に基づく読出し



21.2 論理学とは何か？

- 「論理学は推論の科学である」 — 林 晋
- “Logic deals with what follows from what.”

— Alan Robinson

21.2 さまざまな推論

- 演繹 (deduction)
  - “正しい”(soundな) 推論
- 非演繹的推論
  - 帰納 (induction)
  - アブダクション (abduction)
  - 類推 (analogy)

21.2 演繹の例

$$\frac{\text{human(socrates)} \quad \forall X(\text{human}(X) \Rightarrow \text{mortal}(X))}{\text{mortal(socrates)}}$$

$$\frac{\forall X\,(\text{human}(X) \Rightarrow \exists Y(\text{loves}(X,Y)))}{\text{human(tom)} \Rightarrow \exists Y(\text{loves(tom,Y)})}$$

21.2 帰納の例

buy(a,beer) buy(b,beer) buy(d,beer)
buy(a,snack) buy(b,snack)
buy(c,snack) buy(d,snack)

∀X(buy(X,beer) ⇒ buy(X,snack))

- 知識の一般化によって規則(一般理論)を生成
  - データマイニング，科学的発見
- 与えられた事例以上のことを予測している

21.2 アブダクションの例

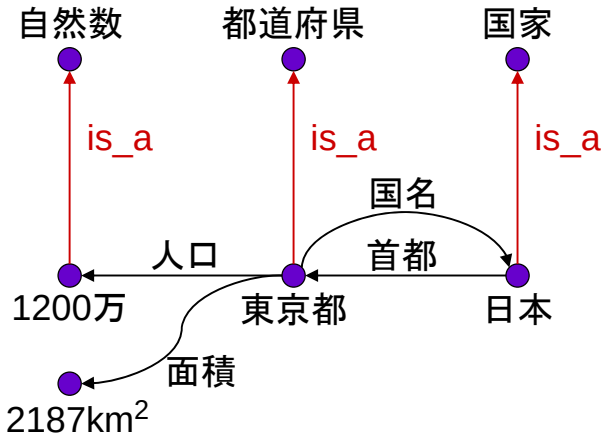
fever(tom)　∀X(cold(X) ⇒ fever(X))

---

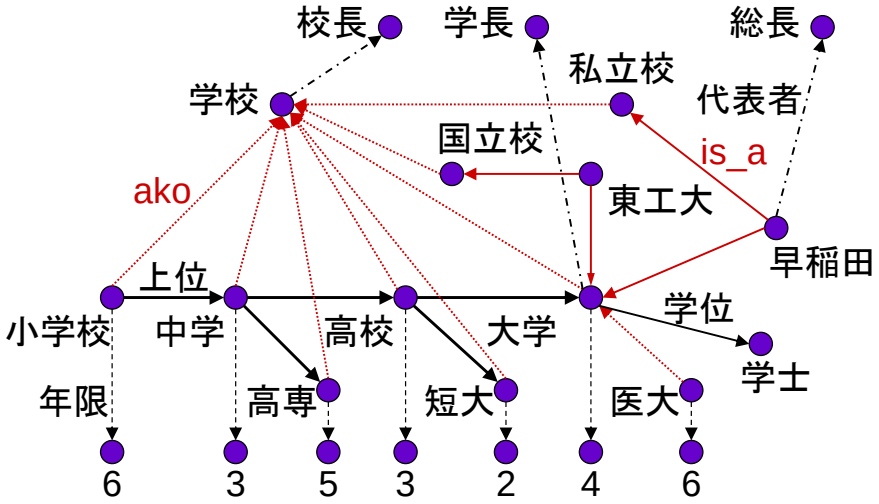
cold(tom)

- 現象と一般理論から(現象の)説明を生成
  - 診断
  - 捜査
- 仮説推論ともいう

22.1 他の知識表現法：Semantic Nets



22.1 他の知識表現法：Semantic Nets



22.1 他の知識表現法：Semantic Nets

- 人間の記憶や連想のモデルとして提案 (1960's)
- リンクは２項関係に対応
- 普遍的な関係がいくつかある
  - instance-class 関係 ("is-a", ∈)
  - class-superclass 関係 ("ako" (a-kind-of), ⊂)
    - 継承 (inheritance)
    - defaults
  - part-whole 関係 ("part-of")

22.1 Semantic Nets と論理式

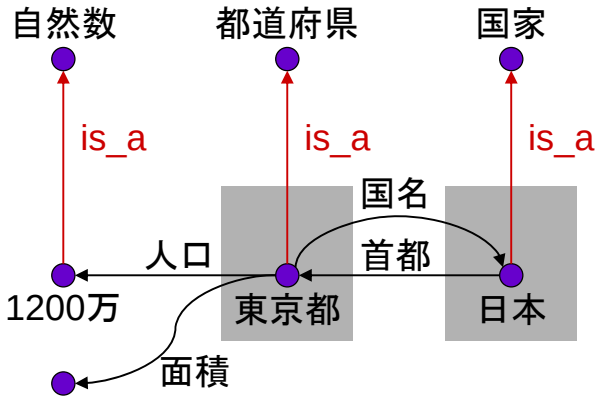
- Semantic nets で表現できることは論理式でも表現できる．逆は工夫が必要.
- 階層化は情報共有の一形態
  - cf. オブジェクト指向言語
    - 差分プログラミング
  - 表現と管理の効率を上げる
  - 非単調推論を可能にする

22.1 非単調推論（参考）

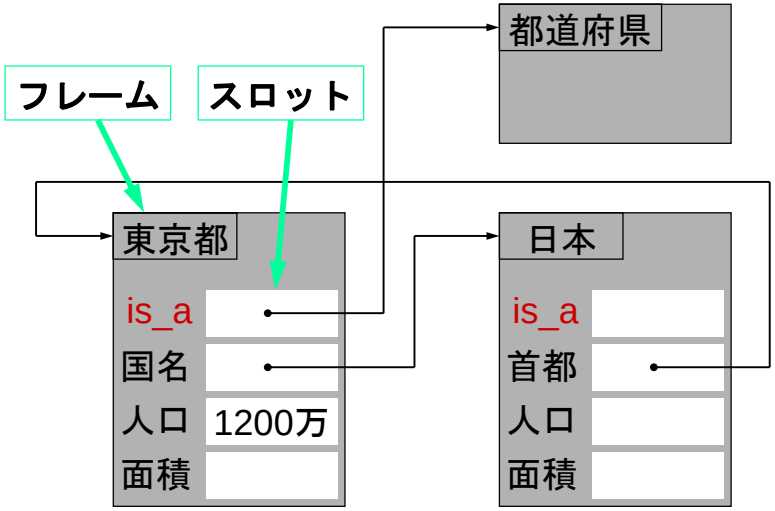
- 述語論理における知識表現と推論は，知識が増えるほど証明できることが増える
  ➔ "単調" という
- 「一般には鳥は飛ぶが，ペンギンやダチョウ（や…）は飛ばない」という知識を扱うには，非単調性を要する
  - 効率よく扱うには、継承とそのキャンセルの機構が有効

22.2 他の知識表現法：フレーム (frames)

- Marvin Minsky, 1970's



22.2 他の知識表現法：フレーム (frames)



22.2 他の知識表現法：フレーム (frames)

- 単なる semantic nets の別表現としてでなく，人間の認知・理解に関する考察から生まれる
- 観察：一度見た／経験した物事の認知過程と，初めての物事の認知過程は大きく異なる
  - 前者は速く，認知的な負担も軽い
- 仮説：トップダウン的な認知
  - 場面全体の記述に適したフレームを取り出し，観測事実に基づいてスロットを埋める
  - 不足情報はデフォールトによって補う

22.2 認知の効率

- 例：
  - 初めて行くレストラン
  - 会合案内や新聞の事故記事の理解
  - 一部が見えない物体の認識
  - 医者の病気診断
  - 出入国管理官との問答
  - 文の理解

23.4 閉論理式 (closed formula)（復習）

- 自由変数を一つももたない論理式を，閉論理式という．
- 平叙文で書かれた知識や数学的性質の多くは，一階述語論理の閉論理式で表現できる（＝知識表現）．
  - 数学では，論理式の頭に並ぶ全称記号がしばしば省略される（例：$x+y=y+x$）
- 知識を論理式で表現するには：
  1. 対象領域（変数の動く範囲）を決める
  2. 適切な定数，関数，述語を導入してその意味を決める
  3. 閉論理式への翻訳を行う

23.4 閉論理式の例（復習）

1. (i) $\exists Y \forall X\, (loves(X,Y))$ (ii) $\forall X \exists Y\, (loves(X,Y))$
   cf. $\exists Y \forall X\, (X>Y)$ $\forall X \exists Y\, (X>Y)$
2. $\forall X \forall L\, (member(X,cons(X,L)))$
   $\wedge \forall X \forall Y \forall L\, (member(X,L) \Rightarrow member(X,cons(Y,L)))$
3. $\forall X\, (fever(X) \wedge cough(X) \Rightarrow cold(X))$
4. $\forall X\, (\neg scolded(X) \Rightarrow \neg study(X))$
5. (i) $\forall X \forall Y\, (X \neq Y \Rightarrow g(X) \neq g(Y))$
   (ii) $\forall Y \exists X \forall X'\, (X' > X \Rightarrow f(X') > Y)$
6. (i) 国籍(夫(花子),米国)
   (ii) $\exists X$ (夫婦(X,花子) $\wedge$ 国籍(X,米国))

23.4 閉論理式を用いた知識表現の練習

1. 花子を知る人はみな彼女を尊敬している
2. 独身の男が孤独とは限らない
3. (i) どんな人にも，母親がちょうど一人いる
   (ii) どんな人にも，親がちょうど二人いる
   （X=Y: X と Y が同一人物である）
4. 最大の整数なるものは存在しない
5. k は m と n の公約数である
6. (i) $\lim_{x\to\infty} f(x) = c$ (ii) $\lim_{x\to a} f(x) = f(a)$
7. $f(n) = O(n^2)$

23.4 閉論理式を用いた知識表現の練習

1. Every son of my father is my brother.
2. Some lecture was attended by every student.
3. Every student attended some lecture.
4. Some student attended every lecture.
5. Every lecture was attended by some student.
6. Some lecture was attended by no student.
7. Only Tom hates Mary.
8. Tom hates Mary only.
9. Tom is as tall as any other classmate of him.

24.1 構造と解釈（復習）

- 一階述語論理式の真偽を論じるには，一般に
  - 対象領域：議論の対象としているモノの世界
  - 記号の意味（＝解釈）：各記号がどのようなモノ・関数・関係を表しているか

  を決めてやる必要がある
- 対象領域＋記号の解釈を構造 (structure) という
- 閉論理式 W の真偽は，一般には，考えている構造に左右されるが，例外的に (?)，いかなる構造の下でも真になる論理式（恒真式）や偽になる論理式（充足不可能式）が存在する．

解釈（復習）

- 対象領域を D とするとき，記号列（論理式やその構成要素） e の解釈は以下のように定まる．
  - e の解釈を $[\![e]\!]$ と書く（outfix 記法）
- n 引数関数記号 f の意味 $[\![f]\!] : D^n \rightarrow D$，および
  n 引数述語記号 p の意味 $[\![p]\!] : D^n \rightarrow \{ false, true \}$
  の意味は人間が決める
  - 例：$[\![1]\!] = 1$，$[\![*]\!](x,y) = x \times y$，$[\![!=]\!](x,y) = x \neq y$
- 項の解釈は $[\![f(t_1, ..., t_n)]\!] = [\![f]\!]([\![t_1]\!], ..., [\![t_n]\!])$
  原子論理式の解釈は $[\![p(t_1, ..., t_n)]\!] = [\![p]\!]([\![t_1]\!], ..., [\![t_n]\!])$
  によって帰納的に定義される．

解釈（復習）

- 論理演算子の意味 $[\![\neg]\!]$，$[\![\wedge]\!]$，$[\![\vee]\!]$，$[\![\Rightarrow]\!]$ は，真理値表が定めるとおり．
  - 例：$[\![\wedge]\!](true, true) = true$，$[\![\wedge]\!](true, false) = false$, . . .
- 限定記号の意味は次のように定める
  - (準備) W の中の自由変数 u の出現を，W 中に出現しない定数 c に置き換えた式を $W[c/u]$ と書く．
  - $[\![\forall u W]\!]$ は「c の解釈，すなわち $[\![c]\!]$ が何であっても $[\![W[c/u]]\!] = true$ になること」と定める
    - 「みんないい人だ」vs.「A さんはいい人だ」
  - $[\![\exists u W]\!]$ は「c の解釈，すなわち $[\![c]\!]$ をうまくとれば $[\![W[c/u]]\!] = true$ になること」と定める

24.2 恒真性・充足可能性（復習）

- 閉論理式はつぎの３種類に分類できる
  - 恒真（注意：「恒真」と「真」とは異なる）：
    - 例： $\forall X(p(X) \vee \neg p(X))$
  - 充足可能だが恒真でない：
    - 例： $\forall X(p(X)) \vee \forall X(\neg p(X))$
  - 充足不可能：
    - 例： $\exists X(p(X) \wedge \neg p(X))$

24.2 論理的帰結（復習）

- 演繹的推論の目的は，証明目標 W が，与えられた前提 $W_1, ..., W_n$ の論理的帰結であることを示すこと．
  - 前提の真偽は問わないが
  - 前提が正しければ結論も正しいことを保証
- W が $W_1, ..., W_n$ の論理的帰結であることと
  $W_1 \wedge ... \wedge W_n \Rightarrow W$ が恒真であることと
  $W_1 \wedge ... \wedge W_n \wedge \neg W$ が充足不可能であること
  は同値
  - ➔ 論理式の充足不可能性の証明に還元可能

25. 論理式の節形式への変換

- 閉論理式の形を型にはめる（cf. 論理積標準形）
- **ステップ１** (25.1)
  - 限量子は木構造の頭の方へ（冠頭標準形）
  - ∧, ∨, ¬ の出方は論理積標準形に合わせる
    - レベル0： ∧
    - レベル1： ∨
    - レベル2：リテラル（原子論理式またはその否定）

基本的な恒真式（復習）

Q は x を自由変数として含まないとする

- $\forall x P \wedge Q \Leftrightarrow \forall x\,(P \wedge Q)$　　$\exists x P \wedge Q \Leftrightarrow \exists x\,(P \wedge Q)$
- $\forall x P \vee Q \Leftrightarrow \forall x\,(P \vee Q)$　　$\exists x P \vee Q \Leftrightarrow \exists x\,(P \vee Q)$
- $\neg \forall x P \Leftrightarrow \exists x\,(\neg P)$　　$\neg \exists x P \Leftrightarrow \forall x\,(\neg P)$
- $\forall x P \wedge \forall x R \Leftrightarrow \forall x\,(P \wedge R)$　　$\exists x P \vee \exists x R \Leftrightarrow \exists x\,(P \vee R)$
- $\forall x P \vee \forall x R \Rightarrow \forall x\,(P \vee R)$　　$\exists x\,(P \wedge R) \Rightarrow \exists x P \wedge \exists x R$
  - 問：最後の二つの式の逆 (⇒ を ⇐ に換えたもの) は恒真でない．次の論理式が偽となる構造を作れ．
    1. ∀X(p(X) ∨ q(X)) ⇒ ∀X(p(X)) ∨ ∀X(q(X))
    2. ∃X(p(X)) ∧ ∃X(q(X)) ⇒ ∃X(p(X) ∧ q(X))

冠頭標準形（復習）

- 論理式の先頭（木構造で考えるときは上の方）に限定記号を集めた論理式．正確には
  - 限定記号を一つも含まない論理式は冠頭標準形
  - 論理式 W が冠頭標準形ならば，論理式 QuW（Q は限定記号，u は変数）も冠頭標準形
- 前頁の公式を使うと，どんな論理式 W も，それと論理的に同値な（つまり $W \Leftrightarrow W'$ が恒真であるような）冠等標準形 W' に変換できる．
  - 問：∀X(∃Y(p(X,Y)) ⇒ ∃Y(p(X,Y) ∧ r(Y))) を冠頭標準形に変換せよ．

25. 論理式の節形式への変換

- **ステップ２** (25.2)
  さらにスコーレム変換で存在限量子を消去

- これによって，すべての論理式は，リテラルの論理和の論理積の形に変換できる (26.3)
  - 論理式は節の集合（論理積）
  - 節はリテラルの集合（論理和）（cf. 確定節）
  - リテラルは原子論理式またはその否定

  ➔ 表記上使う論理記号は ¬ だけ

25. 節形式の性質

全部の論理式を同時に真にするような構造は存在しない

- 空節 { } は偽を表わす（∨ の単位元）
- L と $\neg L$ の両方を含む節 $\{L, \neg L, \ldots\}$ は恒真式
- もとの論理式とその節形式は同値とは限らない（スコーレム変換したため）．しかし

論理式の集合が充足不可能 ⟷ 対応する節集合が充足不可能

したがって，節形式に変換して矛盾を導くことで背理法による証明が構成できる

26. 融合原理

- 節の融合 (resolution) ＝ 三段論法の一般化

$$\frac{\{\neg a, \neg b, d, e, f\} \quad \{\neg e, \neg g, j, k\}}{a \wedge b \Rightarrow d \vee e \vee f \quad e \wedge g \Rightarrow j \vee k}$$

$$a \wedge b \wedge g \Rightarrow d \vee f \vee j \vee k$$

$$\{\neg a, \neg b, \neg g, d, f, j, k\}$$

理由：二つの前提は以下のようにも書ける．

- $\neg e \Rightarrow \underline{\neg a \vee \neg b \vee d \vee f}$
- $e \Rightarrow \underline{\neg g \vee j \vee k}$

よって，下線部の少なくとも一方が成立．

26. 融合原理：命題論理の場合

- 融合節は，もとの二つの節の論理的帰結
- 【注意】下記は誤った推論（複数のリテラルを同時にキャンセルしてはいけない）！

$$\frac{a \wedge b \Rightarrow d \vee e \vee f \quad e \wedge f \Rightarrow j \vee k}{a \wedge b \Rightarrow d \vee j \vee k}$$

— 理由を考えてみよ．

26.1 融合原理による証明

- 融合原理による証明の手順
  1. それぞれの前提を節形式に変換
  2. 結論の否定を節形式に変換
  3. それらから融合によって空節が導かれれば，
     - もとの節集合が充足不可能だった
     - したがって結論が証明された

  ことになる．

26.1 融合原理による証明例

- W_1: Rain ⇒ Wet　　C_1: {¬Rain, Wet}
- W_2: ¬Rain ⇒ Swim　C_2: {Rain, Swim}
- W_3: Swim ⇒ Wet　　C_3: {¬Swim, Wet}
- ¬W_4: ¬Wet　　C_4: {¬Wet}

26.1 融合原理による証明例

- W_1: Rain ⇒ Wet　　C_1: {¬Rain, Wet}
- W_2: ¬Rain ⇒ Swim　C_2: {Rain, Swim}
- W_3: Swim ⇒ Wet　　C_3: {¬Swim, Wet}
- ¬W_4: ¬Wet　　C_4: {¬Wet}

- C_1 と C_4 から　　C_5: {¬Rain}
- C_2 と C_5 から　　C_6: {Swim}
- C_3 と C_6 から　　C_7: {Wet}
- C_4 と C_7 から　　C_8: { } (矛盾)
- よって $W_1, W_2, W_3 \models W_4$ が示された.

27. 融合原理：一階述語論理の場合

- 融合時にキャンセルする正負のリテラル対の（引数の）形を揃える必要がある
  ➔ 単一化 (unification) によって，変数がどう化けるべきかを求める
- 例：

$$\frac{\text{human(socrates)} \quad \forall X(\text{human}(X) \Rightarrow \text{mortal}(X))}{\text{mortal(socrates)}}$$

では human(socrates) と human(X) を単一化して，右側の一般的知識をまず具体的な形にする

Prologの実行メカニズム（復習）

- 形を揃える操作 ＝ 単一化 (unification)
  例：human(socrates) と
  human(X) ⇒ mortal(X) の左辺は，
  X ← socrates と化ければ同じ形になる
- Prolog は，プログラミング言語に必要な
  - 代入（assignment）機構
  - 同一性の比較機構
  - 引数の受渡し機構

  をすべて単一化によって実現している

27.1 代入と単一化

- 代入 (substitution)：変数の化けかたを指定したもの（表記法：$\{v_1 \leftarrow t_1, \ldots, v_n \leftarrow t_n\}$）
  - $v_1,\ldots,v_n$ は互いに異なる変数
  - t_k は v_k とは異なる
  - ベキ等代入：化けてなくなる変数が新たに持ち込まれることがない代入
- 単一化：二つの項の変数を適当に化けさせて，両者を同じ形にすること
  - 最小限の化け方で同じになるのがうれしい
    ➔ 最汎ユニファイア (**m**ost **g**eneral **u**nifier)

27.1 方程式の変形による単一化

(a) $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)$ ➔ $s_1 = t_1, \ldots, s_n = t_n$
(b) $f(s_1,\ldots,s_n) = g(t_1,\ldots,t_m)$ ➔ 失敗して終了
(c) $x = x$ ➔ 等式を消す
(d) $t = x$ (t は変数でない) ➔ $x = t$ に変形
(e) $x = t$ ➔ 他の式の中の x を全部 t に書換え
($t \neq x$, x は t 中に現れない)
(f) $x = t$ ➔ 失敗して終了
($t \neq x$, x は t 中に現れる)

上記の操作を繰り返す．すべての式が (e) の形になったら，「解けた」ことになる．

問70： p(f(X,k),Y,Y) = p(A,g(A),g(B))

- (a) から
  f(X,k)=A
  Y=g(A)
  Y=g(B)
- (d) から
  A=f(X,k)
  Y=g(A)
  Y=g(B)
- (e) から
  A=f(X,k)
  Y=g(f(X,k))
  Y=g(B)

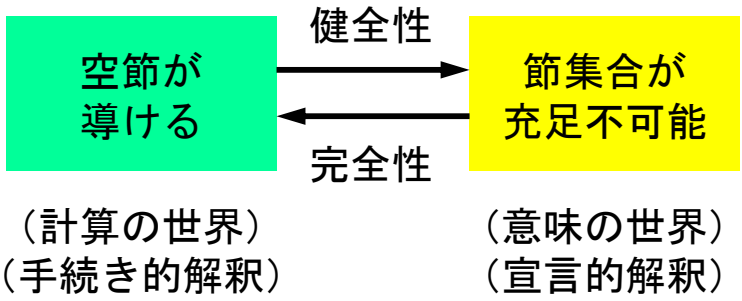
- (e) から
  A=f(X,k)
  g(B)=g(f(X,k))
  Y=g(B)
- (a) から
  A=f(X,k)
  B=f(X,k)
  Y=g(B)
- (e)から
  A=f(X,k)
  B=f(X,k)
  Y=g(f(X,k))

27.2 融合原理：一階述語論理の場合

- 命題論理の場合と異なるのは，引数を揃える作業（＝ 単一化）を伴う点
- 次の二つのことに注意すれば，命題論理の場合の自然な拡張で証明できそう
  - 節の具体例（＝ 変数を具体化したもの）は，もとの節の論理的帰結になっている
  - キャンセルしようとする原子論理式の引数の形がすでに揃っているならば，そのままキャンセルして融合節を作ればよい

27.2 融合原理は強力

- 融合原理は健全かつ完全



27.3 融合原理による証明の例

- ここでいくつかの例を紹介

27.3 穴埋め型質問への答え方

- Prolog のゴールは条件文の略記とも解釈できた.

  yes(X) :- is_ancestor_of(X, jim)

  = If X is an ancestor of jim, say “yes” and answer X.
- yes(...) を補った場合は，節の中が yes(...) だけになるまで融合を行えばよい.

28. 確定節 (definite clause)

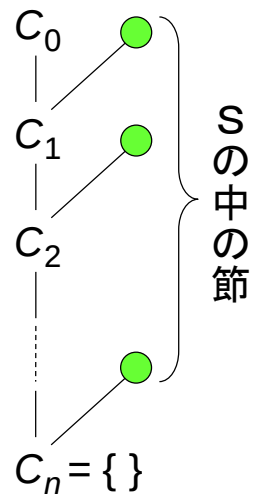
- $\{L, \neg L_1, \neg L_2, \ldots, \neg L_n\}$ の形の節（$n \geq 0$）
  （cf. Prolog の規則）
- Prolog の質問も以下の形に直せば確定節
  - yes(X) :- is_ancestor_of(X, jim)
- 多くの知識は確定節を用いて書ける．ただし
  - 例外1:「～ でない」
  - 例外2:「～ と ～ は両立しない」「～ だったら ～ ではない」（一貫性制約）
  - 例外3:「～ または ～ だ」

28. 確定節の重要な特徴

- 確定節と確定節を融合してできる節は確定節
- したがって，確定節の集合は決して矛盾しない（＝充足可能）．
  - ∵ もし矛盾するならば，融合の完全性から空節が導けるはず．

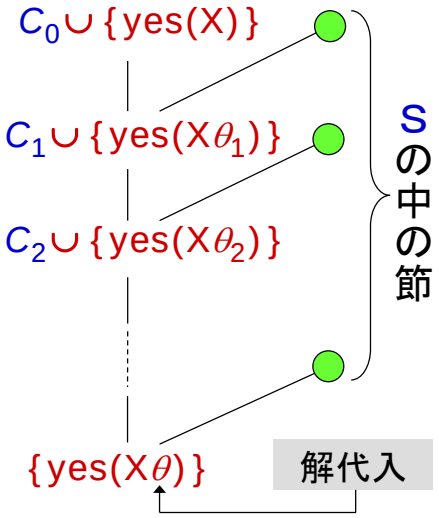
28. SLD融合による証明

- S: 確定節の集合（cf. Prologプログラム）
- C_0: 負のリテラルだけからなる節（cf. Prologのゴール）
- $S \cup \{C_0\}$ が充足不可能ならば，必ず右の形の（＝線形の）証明がある．しかも各ステップでは，C_i 中の好きに選んだゴールの消去を考えるだけでよい



28. SLD融合の特徴 (1)

- 答が確定的．質問を $C_0 \cup \{\mathrm{yes(X)}\}$ の形で与えた場合，答があれば必ず $\{\mathrm{yes(X}\theta\mathrm{)}\}$ の形になり，$\{\mathrm{yes(X}\theta\mathrm{), yes(X}\theta'\mathrm{)}\}$ とはならない



28. SLD融合の特徴 (2)

- 質問が１個の原子論理式 g の場合を考える．計算された答（＝ 解代入）をもとの質問 g に代入したものは，与えた確定節集合 S の論理的帰結になっている．
  - ∵ S と $\{\neg g\}$ から空節を導くことができるならば，S と $\{\neg g, g\}$ から同じ道筋の証明を構成すると，$\{g\theta\}$（θはこの証明の解代入）が残る．これは $\forall(g\theta)$ を意味していて，しかも S の論理的帰結である．