論理と計算

第７回

述語論理：推論

担当：尾崎　知伸

ozaki.tomonobu@nihon-u.ac.jp

講義予定　※一部変更（前倒し）になる可能性があります

09/22	01. オリエンテーション と 論理を用いた問題解決の概要
09/29	02. 命題論理：構文・意味・解釈
10/06	03. 命題論理：推論
10/13	04. 命題論理：充足可能性問題
10/20	05. 命題論理：振り返りと演習（課題学習）
10/27	06. 述語論理：構文・意味・解釈
11/03	07. 述語論理：推論　※文化の日，文理学部授業日
11/10	08. 述語論理：論理プログラムの基礎
11/17	09. 述語論理：論理プログラムの発展
11/24	10. 述語論理：振り返りと演習（課題学習）
12/01	11. 高次推論：発想推論
12/08	12. 高次推論：帰納推論の基礎
12/15	13. 高次推論：帰納推論の発展
12/22	14. 高次推論：振り返りと演習（課題学習）
01/19	15. まとめと発展的話題

目次：今回の授業の内容

- 述語論理の標準形
  - 冠頭標準形
  - スコーレム標準形
  - 節形式

- 述語論理の推論
  - 代入と最汎単一化
  - 融合法
  - 融合法の戦略

命題論理から一階述語論理へ

- 命題論理（propositional logic）
  - 「命題」を基本要素とする論理言語 → 命題のみしか扱うことができない
  - 表現能力に乏しい場合も．．
    - 多くのオブジェクトが存在する場合に，それを簡潔に記述することができない
      - 例：「穴のある部屋の隣では風を感じる」in Wumpus World
        ルールは一つだが，実際には，各部屋についてそれぞれ記述する必要がある．
    - オブジェクト間の関係を簡潔に記述することができない
      - 例：親子関係，各親子関係についてバラバラに命題を準備しなければならない

- 命題論理の基本要素（文脈非依存，無曖昧性，宣言的など）を土台に表現力を豊かにする
  - 自然言語における統語論から．．．．
  - 「オブジェクト」(object)に相当する名詞・名詞句（部屋，穴など）
  - 「オブジェクト間の関係」（relation）に相当する動詞・動詞句（隣接している）
    - 関係の一種としての単項関係＝性質
    - 関係の一種としての「関数」(function)：入力に対して唯一の値が存在する関係

- 述語論理（predicate logic）
  - 述語論理：「オブジェクト」と「関係」を基本要素とする論理言語
  - 一階述語論理（first-order logic）：オブジェクトに対する量化（変数化）を許す
  - 二階述語論理（second-order logic）：述語や関数記号に対する変数化を許す
    - 高階述語論理（higher-order logic）

述語論理（predicate logic）

- 述語論理の特徴
  - 世界に登場する「個々の対象」を表現できる
  - 対象と対象の関係を中心に考え，個々の対象間の「関係」を述語を用いて表現する
  - 無限の対象からなる世界を表現できる
  - 準決定的である（正しい文は反駁証明可能．正しくない文は，有限時間で証明できるとは限らない）
- 述語論理の構成要素
  - 述語文・述語論理式
  - 対象と述語（対象の性質や対象間の関係）の2種
    - 対象：文中に現れる主語や目的語に相当 / 変数にしても良い
    - 述語：動詞や形容詞に相当
      - "述語（対象，対象．．．．）"の形式で記述します．
      - orbits(earth, sun) / parent(tom, jack) / fever(ann)
      - catch_a_cold(X)⇒have_a_cough(X)
  - 命題論理と同様，結合子を用いて複合文を構成する
  - 限量子（quantifier）
    - 全称限量子（universal quantifier）∀：「すべてのXに対して～」
    - 存在限量子（existential quantifier）∃：「あるXに対して～」
    - ∀X ∃Y child(X, Y)

一階述語論理の構文（Syntax）

命題論理との違いに着目しよう

文	→	原子文 | 複合文
原子文	→	述語記号 | 述語記号 (項,···) | 項 = 項
複合文	→	（文）
	|	¬ 文
	|	文 ∧ 文
	|	文 ∨ 文
	|	文 ⇒ 文
	|	文 ⇔ 文
	|	限量子 変数, ··· 文
項	→	定数
	|	変数
	|	関数記号 (項,···)
限量子	→	∀ | ∃
変数	→	大文字で始まる文字列
定数	→	小文字で始まる文字列 | 数値
関数記号	→	小文字で始まる文字列
述語記号	→	true | false | 小文字で始まる文字列

結合力（降順）：¬, ∧, ∨, ⇒, ⇔

項（Term）：オブジェクトの論理表現

変数（Variable）：オブジェクトの抽象表現

限量子（quantifier）：量化（変数化）

解釈

命題論理に帰着
基礎原子文＝命題

- 領域：述語文 α が表現する世界での対象の集合

- 解釈：述語文 α の領域をDとする．このとき， α の解釈を以下のステップで与える
  1. α 中の定数記号，関数記号への割り当て
     - 定数記号c にDの元c^Iを割り当てる
     - n引数の関数記号fにD上の関数：$f^I:D^n \rightarrow D$を割り当てる
  2. 基礎原子文への真理値の割り当て
     - 上記の割り当てにより， α 中のすべての述語に対して，それらの基礎原子文がすべて決まる．それらの各基礎原子文に対して，真理値 {true, false} を割り当てる
     - すなわち，n引数の述語記号pにD上の関数：$p^I:D^n \rightarrow$ {true, false} を割り当てる

- モデル： α を真にする解釈Iを， α のモデルと呼ぶ
  - 文 α の真理値の決定
    - 基礎原子文の真理値は，解釈で直接与えられる
    - 複合文の真理値は，結合子（¬，∧，∨，⇒，⇔）の真理値表から計算
    - ∀X (P)の真理値は，D中のすべての要素xについてPがtrueならばtrue
      そうでなければfalse
    - ∃X (P)の真理値 は，D中の少なくとも一つの要素xについてPがtrueならばtrue
      そうでなければfalse

述語論理文の分類と論理的帰結

- 命題論理と同様，3種類に分類
  - 恒真：すべての解釈のもとで真となる文
  - 恒偽：すべての解釈のもとで偽となる文
    - 矛盾（contradiction），充足不能（unsatisfiable）とも呼ぶ
  - 充足可能：文が真となる解釈が存在する

- 論理定数true は恒真である
- 論理定数falseは恒偽である．
- 任意の基礎原子文 α は充足可能である

- 定理：$G = \{\gamma_1, \gamma_2, \ldots\}$を述語文の集合，$\alpha$を述語文とする．伴意式$G \models \alpha$が成り立つのは，$(\gamma_1 \wedge \gamma_2 \wedge \cdots) \Longrightarrow \alpha$が恒真のとき，かつ，そのときのみである
  - $(\gamma_1 \wedge \gamma_2 \wedge \cdots) \Longrightarrow \alpha$が恒真 $\Leftrightarrow$ $(\gamma_1 \wedge \gamma_2 \wedge \cdots \neg \alpha)$が恒偽

- 恒真（恒偽）のチェックは，命題論理の場合以上に困難
  - すべての世界，すべての解釈について調べることは困難 → 証明論が必要
    - ※対象を限定し，陽に解釈・モデルを考えるのもありかとは思います

述語論理：標準形

述語論理の標準形と節集合

- 同じ内容を表す述語論理文は複数考えられる
  - ∀X(human(X) ⇒ ¬connect(chin(X), elbow(X))：すべての人間は，顎と肘をつけられない
  - ¬ ∃X（human(X)∧connect(chin(X),elbow(X))：顎と肘がつく人間はいない
  - 形を限定する → 議論がしやすい．効率的な推論が構築できる．

リテラル
命題論理：命題記号とその否定
述語論理：原子式とその否定

- （復習）命題論理の標準形
  - 連言標準形（Conjunctive Normal Form）：選言の連言（「リテラルの∨」の∧）
    - 節形式（Clausal Theory）とも呼ばれる．DPLLなどSAT問題における標準的な入力形式
  - 選言標準形（Disjunctive Normal Form）：連言の選言（「リテラルの∧」の∨）

- 述語論理の標準形
  - **冠頭標準形**（Prenex Normal Form）：式の左端（先頭）ですべての変数が限量された述語文
  - **スコーレム標準形**（Skolem normal Form）：
    - 「母式が連言標準形」である「存在限量子を含まない」冠頭標準形の述語文
  - **節集合**（Clause Set）：0個以上の節（clause）の連言
    - 節（clause）：すべての変数が全称限量（∀）されたリテラルの0個以上の選言
      - 節集合は，存在限量（∃）を含まない（∃を消す変換をする）
    - 節集合は（述語論理における）融合法が前提とする形

∀X (.. ∨ .. ∨ ..)
∧ . . . ∧
∀Z (.. ∨ .. ∨ ..)

節集合：
「それぞれ全称限量」された節
（＝リテラルの∨）の∧

- 述語論理文の節集合への変換（融合法が利用できる形式への変換）
  - 文 → 冠頭標準形 → スコーレム標準形 → 節集合 の順に変換

冠頭標準形

- 定義：Q_iを∀あるいは∃として，すべての変数が式の左端（先頭）で限量された$(Q_1)\cdots(Q_n)(M)$の形をした述語文
  - このとき，Mを母式（matrix）と呼ぶ
- 要は「すべての限量子が文の先頭にある閉じた文」ということ
- 例
  - ∃Y ∀X (p(X, Y))　　# 母式： p(X, Y)
  - ∀X ∀Y (parent(X, Y) ⇒ child(Y, X))　　# 母式： parent(X, Y) ⇒ child(Y, X)
  - ∀X ∃Y (¬p(X) ∨ (q(Y) ∧ r(X, Y)))　　# 母式： ¬p(X) ∨ (q(Y) ∧ r(X, Y))
- 次ページの変換手続きを用いることで，与えられた（閉）論理式を冠頭標準形に変換可能

冠頭標準形への変換

- 以下の変換手続きを用いることで，与えられた（閉）論理式を冠頭標準形に変換可能

1. 以下の等式を用いて，含意記号（⇒）と同値記号（⇔）とを除去する
   1. $(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$
   2. $(p \Leftrightarrow q) \Leftrightarrow ((p \Rightarrow q) \wedge (q \Rightarrow p))$

2. 以下の二重否定のトートロジーと，ド・モルガンの法則を用いて否定記号を原子文の直前に移動する
   1. $\neg(\neg \alpha) \Leftrightarrow \alpha$
   2. $\neg(\alpha \wedge \beta) \Leftrightarrow \neg \alpha \vee \neg \beta$
   3. $\neg(\alpha \vee \beta) \Leftrightarrow \neg \alpha \wedge \neg \beta$
   4. $\neg \exists X \alpha(X) \Leftrightarrow \forall X \neg \alpha(X)$
   5. $\neg \forall X \alpha(X) \Leftrightarrow \exists X \neg \alpha(X)$

3. 以下の等式を用いて，限量子を左端に移動する
   1. $QX(\alpha\{X\}) \vee \beta \Leftrightarrow (QX)(\alpha\{X\} \vee \beta)$　#Qは限量子，$\alpha\{X\}$はXを含む述語文，βにはXは現れない
   2. $QX(\alpha\{X\}) \wedge \beta \Leftrightarrow (QX)(\alpha\{X\} \wedge \beta)$　#限量子のスコープを変えている
   3. $\forall X(\alpha\{X\}) \wedge \forall X(\gamma\{X\}) \Leftrightarrow \forall X(\alpha\{X\} \wedge \gamma\{X\})$　# 全称限量で∧
   4. $\exists X(\alpha\{X\}) \vee \exists X(\gamma\{X\}) \Leftrightarrow \exists X(\alpha\{X\} \vee \gamma\{X\})$　# 存在限量で∨
   5. $\forall X(\alpha\{X\}) \vee \forall X(\gamma\{X\}) \Leftrightarrow \forall X(\alpha\{X\}) \vee \forall Z(\gamma\{Z\})$ #全称限量で∨，変数の改名(renaming)
   6. $\exists X(\alpha\{X\}) \wedge \exists X(\gamma\{X\}) \Leftrightarrow \exists X(\alpha\{X\}) \wedge \exists Z(\gamma\{Z\})$ #存在限量で∧，変数の改名(renaming)

スコーレム標準形

- 定義：「母式が連言標準形」である「存在限量子を含まない」冠頭標準形の述語文
- 冠頭標準形からスコーレム標準形へ…
  → 存在限量子の削除・連言標準形への変換

- 存在限量子の削除 ：存在限量されている変数を「特別な定数」に置き換える
  - 存在限量されている変数が存在しない → 存在限量子は不要で削除できる

- **スコーレム化**：スコーレム定数・関数を導入して存在限量変数を除去する処理
  - 存在限量された変数は「何かあるもの」
  - この「何か」を表す特別な変数 → **スコーレム定数**（Skolem constant）

  - 全称限量のスコープ内に（全称限量変数と共に）現れる存在限量変数の場合．．．．
    - 「何か」は全称限量変数に依存する．
      - ∀X ∃Y loves(X, Y) ："それぞれのXに対して" Yが存在する（YはXの関数）
    - → 全称限量変数に対する関数（xを引数とする関数記号） → **スコーレム関数**（Skolem function）

- スコーレム化後に，母式を連言標準形に変換する

- 例１：∃Y ∀X p(X, Y)： Yをスコーレム定数 aで置き換えて ∀X p(X, a)
- 例２：∀X ∃Y p(X, Y)： YをXのスコーレム関数h(X)で置き換え∀X p(X, h(X))

節集合

- スコーレム標準形： 母式が連言標準形である存在限量子を含まない冠頭標準形の述語文
  - 文全体が，左端で全称限量された連言標準形
- 節集合：「すべての変数が左端で全称限量された節」の連言

  →スコーレム標準形において左端に集められている全称限量子を
  各節（連言子）へ分散させればよい.

- 例：$\forall X (animal(X) \Rightarrow \exists Y (heart(Y) \wedge has(X, Y)))$　すべての動物は心臓を持つ
- 冠頭標準形へ
  - [含意記号の除去]　$\forall X (\neg animal(X) \vee \exists Y (heart(Y) \wedge has(X, Y)))$
  - [存在限量子の移動]　$\forall X \exists Y (\neg animal(X) \vee (heart(Y) \wedge has(X, Y)))$
- スコーレム標準形へ
  - [スコーレム関数の導入] $\forall X (\neg animals(X) \vee (heart(h(X)) \wedge has(X, h(X))))$
  - [母式を連言標準形へ]
    - $\forall X ((\neg animals(X) \vee heart(h(X))) \wedge (\neg animals(X) \vee has(X, h(X))))$
- 節集合へ
  - [限量子の分配と変数の改名]
    - $\forall X (\neg animals(X) \vee heart(h(X))) \wedge \forall Z(\neg animals(Z) \vee has(Z, h(Z)))$
    - 集合として表現すると$\{\neg animals(X) \vee heart(h(X)), \neg animals(Z) \vee has(Z, h(Z)) \}$

代入と最汎単一化

論理的帰結と推論

- そもそもやりたいこと：伴意関係$G \models \alpha$が成り立つかを調べる
  - 文集合Gのもとで，どんな文αが論理的に成り立つかを知りたい
    - 既知の知識から，どんな知識や事実が導かれるのか知りたい
  - 文αがGのもとで，論理的に正しいかを知りたい
    - 新しく考えた知識や事実が，既知の知識と矛盾しないか知りたい

- モデル論的アプローチ：原理的には，すべてのモデル（解釈）を考えればよい
  → 現実的には困難（$2^{\text{基礎アトムの個数}}$ の解釈が考えられる）

- 証明論的アプローチ
  - 理論G＋公理から，いくつかの文を選択し，推論規則を適用することで，新たな文を生成する．この過程を繰り返すことによって，目的とする文αを導く
  - 推論式：$G \vdash \alpha$「理論Gから文αが推論される（証明される）」
    - 完全性，健全性には注意が必要

- 融合法：述語論理における推論方法の一つ
  - 命題論理の場合と同様，健全かつ反駁完全（※融合法＋包摂演算だと健全かつ完全）
  - 融合法では，述語論理文が「節集合」で表現されていることを前提とする

融合法に向けて：代入

- 命題論理における融合法：相補リテラルを持つ節同士を融合（∨で連結）する
- 述語論理でも考え方は同じ：変数の扱いが必要 → 「代入」と「単一化」

- 代入（substitution）
  - 全称限量変数を「別の項に置き換える」処理
    - 変数を，定数や別の変数，関数記号を含む項に置き換えることを代入と言う
    - 全称限量変数を含む文は，（全称限量なので）その変数に何を代入しても成り立つ
  - 代入は，"変数 / 項" の集合で与える．
  - $\alpha\ \theta$ ：文 α に代入 θ を適用した結果得られる文
    - 例：文 α = loves(X, Y)，代入 θ = { X/tom, Y/ann }のとき， $\alpha\ \theta$ は loves(tom, ann)になる

- 基礎代入（ground substitution）と基礎例（ground instance）
  - 文 α に対し， $\alpha\ \theta$ が変数を含まない代入 θ を，（α に対する）基礎代入と呼ぶ．
  - またこのとき得られる $\alpha\ \theta$ を基礎例と呼ぶ

- 合成代入
  - $\Theta = \{U_1/s_1, \cdots, U_m/s_m\}$, $\phi = \{V_1/t_1, \cdots, V_n/t_n\}$をそれぞれ代入とする．
  - Θ 中の各s_iに出現する変数V_jに代入$\{V_j/t_j\}$を施した上で，さらに θ 中に出現しなかった ϕ 中の代入を加えたものを θ と ϕ の合成代入$\theta \circ \phi$と呼ぶ
  - なお$(\alpha\theta)\phi = \alpha(\theta \circ \phi)$が成り立つ（文 α に順番に代入しても，合成代入を代入しても結果は同じ）

融合法に向けて：単一化

- 単一化（unification）とは
  - 要素数2以上の原子文集合に対し，各原子文に代入を施し，それらをすべて同じにする操作
  - この授業では，2つの原子文に対する単一化のみを考える

- 単一化代入(Unifier)：文s と 文tに対し，$s\theta = t\theta$ となる代入を，単一化代入（単一化子）と呼ぶ
  - 単一化可能（不能）：二つの原子文を同一にすることができる（できない）場合を指す
    - ※単一化可能な条件を考えてみよう（具体的なアルゴリズム・実装は後述）
  - 単一化操作は，単一化「不能」の場合は，「失敗」する
  - 単一化操作は，単一化「可能」の場合は，「単一化代入」を返す
    - 実際には，単一化代入は複数考えられるので，最汎単一化代入を返す
  - 例：{ p(X,Y)，p(0, s(0)) } の単一化代入：{ X/0, Y/s(0) }, {X/0, Y/s(Z), Z/0 } …
  - 例：{ p(X,Y)，p(s(A), B)} の単一化代入：{X/s(A), Y/B}, {X/s(a), Y/b，A/a, B/b } …
  - 例：{ p(X, s(0)), q(0,Y) }は単一化不能．{ p(0, s(0)), p(Y,Y) }も単一化不能

- 最汎単一化代入（most general unifier, MGU）：最も一般的な項を返す単一化代入
  - 直感的には，「必要以上の具体化を行わない単一化代入」のこと
  - 定義：n個の表現（項・文）$A_1,\cdots,A_n$に単一化代入θ_0が存在し，$A_1,\cdots,A_n$に対する他の単一化代入θに対し，$\theta_0 \circ \phi = \theta$となる代入$\phi$が存在するとき，$\theta_0$を$A_1,\cdots,A_n$の最汎単一化代入という

  - 例：{ p(X, s(0)), p(0, Y) } のMGU：{ X/0, Y/s(0) }
  - 例：{ p(X, s(0))，p(Y, Y) }のMGU：{X/s(0), Y/s(0) }
  - 例：{ p(X, g(X,Y)), p(Z, g(a, f(Z))) }のMGU：{ X/a, Y/f(a), Z/a}

最汎単一化代入を求めるアルゴリズム

- 表現Aと表現Bの最汎単一化代入を求めるアルゴリズム

1. θ ＝{ } で初期化する
2. 表現A，Bを左から順に調べ，初めて異なる表現が現れたときの各記号からなる2要素の集合を求める．この集合を，不一致集合と呼ぶ．
3. 不一致集合が空集合であれば，（表現A,B,は同一であるので）解θを返して終了
4. 不一致集合中の要素で，片方が変数Vで他方が項Tであれば，代入V/Tをθに追加する．そのような要素がない場合（すなわち両方ともが定数である場合）は，単一化失敗として終了
5. A＝A{V/T}，B＝B{V/T}（すなわちA,Bに代入{V/T}を施す）として，ステップ2に戻る

- 参考：Prologを使えば，MGUを簡単に求めることができます．
  - ?- A = B.
  - ?- unify_with_occurs_check(A, B).
  - ?- unifiable(A, B, MGU).　#swi-prologの場合

  - Swi-prolog : https://www.swi-prolog.org/
  - GNU Prolog：http://www.gprolog.org/
  - Yap-prolog : https://github.com/vscosta/yap-6

参考　　S. Russell, P. Norvig 著，古川康一監訳：エージェントアプローチ人工知能，9章より

単一化アルゴリズム

function UNIFY(x, y, θ=*empty*) **returns** a substitution to make x and y identical, or *failure*
  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(ARGS(x), ARGS(y), UNIFY(OP(x), OP(y), θ))
  else if LIST?(x) **and** LIST?(y) **then**
    return UNIFY(REST(x), REST(y), UNIFY(FIRST(x), FIRST(y), θ))
  else return *failure*

function UNIFY-VAR(var, x, θ) **returns** a substitution
  if $\{var/val\} \in \theta$ for some val **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ for some val **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** *failure*
  else return add $\{var/x\}$ to θ

Figure 9.1 The unification algorithm. The arguments x and y can be any expression: a constant or variable, or a compound expression such as a complex sentence or term, or a list of expressions. The argument θ is a substitution, initially the empty substitution, but with $\{var/val\}$ pairs added to it as we recurse through the inputs, comparing the expressions element by element. In a compound expression such as $F(A, B)$, OP(x) field picks out the function symbol F and ARGS(x) field picks out the argument list (A, B).

融合法

融合（resolution）

- 基本は，命題論理における融合と同じ．変数に対し，単一化が必要
- 前提：節集合を対象とする．各節はリテラルの∨だが，これを集合と見做す

- ２つの節$C_1 = D_1 \cup \{l_1\}, C_2 = D_2 \cup \{\neg l_2\}$に対し，$l_1$と$l_2$が単一化可能でそのMGUを$\theta$とする
  - このような節の対を「融合可能な節の対」と呼ぶ
  - 融合：C_1とC_2から節$C = D_1\theta \cup D_2\theta$を求める操作
  - 融合節：融合で得られる節C

- 融合証明手続き：（以下で与えられる）節集合Σ_0から節αを導出する手続き
  1. $\Sigma = \Sigma_0$とする
  2. Σ中から融合可能な節の対を選び，融合を行い融合節Cを得る
  3. もし$C = \alpha$なら停止する（導出完了）
  4. $\Sigma = \Sigma \cup \{C\}$としてステップ2に戻る
- 融合証明手続きによってΣ_0からαが導出されることを，$\Sigma_0 \vdash_r \alpha$と表記する

- 例：{ ¬ f(X, Y) ∨ p(X, Y), ¬ p(A, B) ∨ c(B, A), f(tom, jane) } からc(jane, tom)を導出する

$$\dfrac{\dfrac{\neg f(X,Y) \vee p(X,Y) \quad \neg p(A,B) \vee c(B,A)}{\neg f(A,B) \vee c(B,A)}\ \{X/A, Y/B\} \qquad f(tom, jane)}{c(jane, tom)}\ \{A/tom, B/jane\}$$

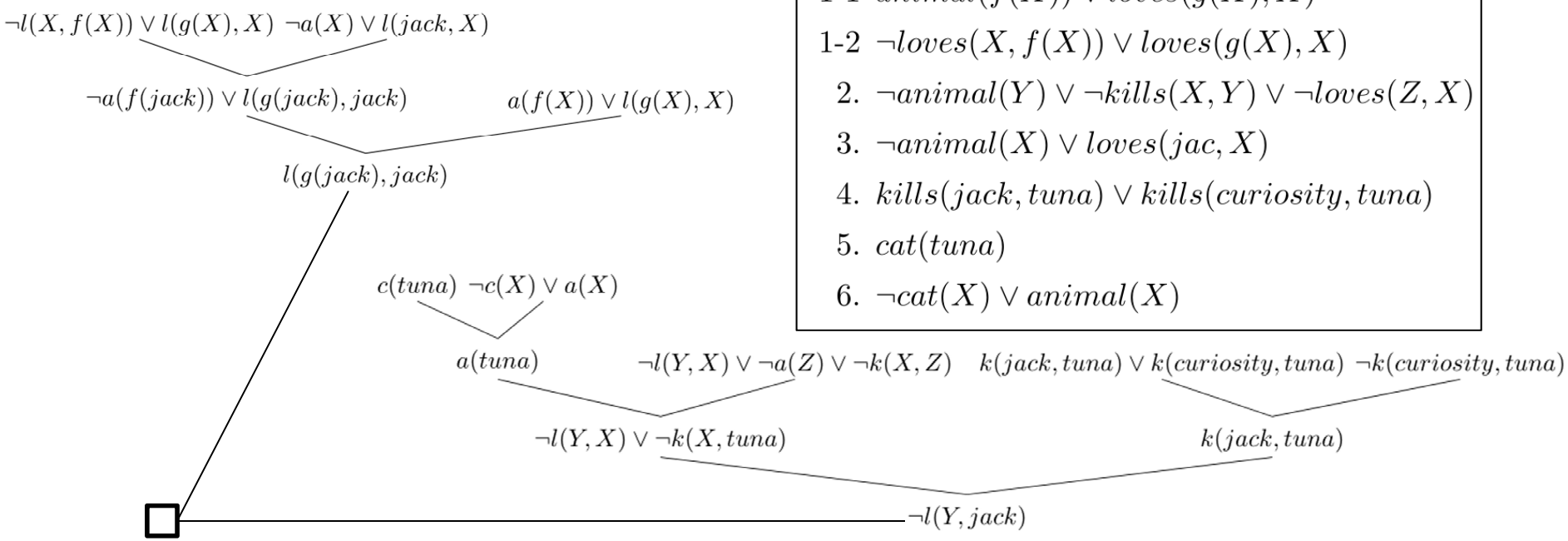
- 以下の条件の下で「Curiosityはその猫を殺したか？」を反駁証明する

- 動物が好きな人は，誰かに愛される．
- 動物を殺す人は誰にも愛されない．
- Jackはすべての動物を愛する
- JackかCuriosityのどちらかが
  Tunaという名前の猫を殺した
- Tunaは猫である．
- 猫は動物である

0. $\neg kills(curiosity, tuna)$
1. $\forall X\,(\forall Y\,(animal(Y) \Rightarrow loves(X,Y)\,) \Rightarrow \exists Z\,(loves(Z,X)\,)\,)$
2. $\forall X\,(\exists y\,(animal(Y) \wedge kills(X,Y)\,) \Rightarrow \forall Z\,(\neg loves(Z,X)\,)\,)$
3. $\forall X\,(\ animal(X) \Rightarrow loves(jack,X)\,)$
4. $kills(jac, tuna) \vee kills(curiosity, tuna)$
5. $cat(tuna)$
6. $\forall X\,(cat(X) \Rightarrow animal(X)\,)$

節集合へ変換

0. $\neg kills(curiosity, tuna)$
1-1 $animal(f(X)) \vee loves(g(X), X)$
1-2 $\neg loves(X, f(X)) \vee loves(g(X), X)$
2. $\neg animal(Y) \vee \neg kills(X,Y) \vee \neg loves(Z,X)$
3. $\neg animal(X) \vee loves(jac, X)$
4. $kills(jack, tuna) \vee kills(curiosity, tuna)$
5. $cat(tuna)$
6. $\neg cat(X) \vee animal(X)$

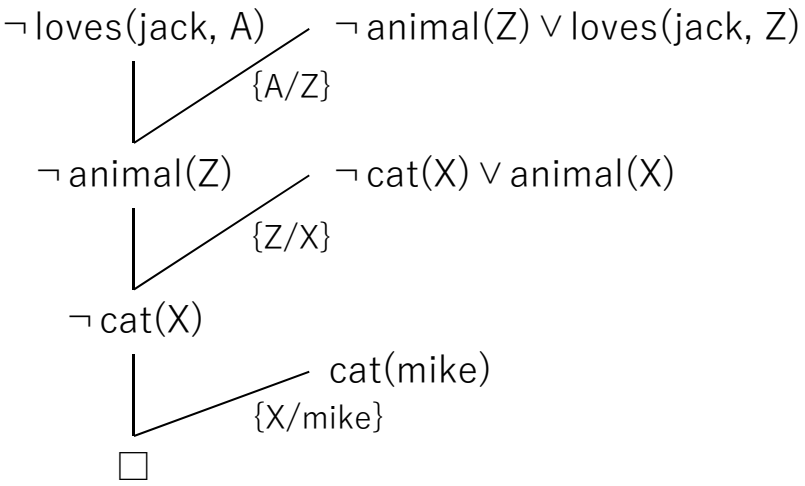


特殊疑問文（What型の質問）

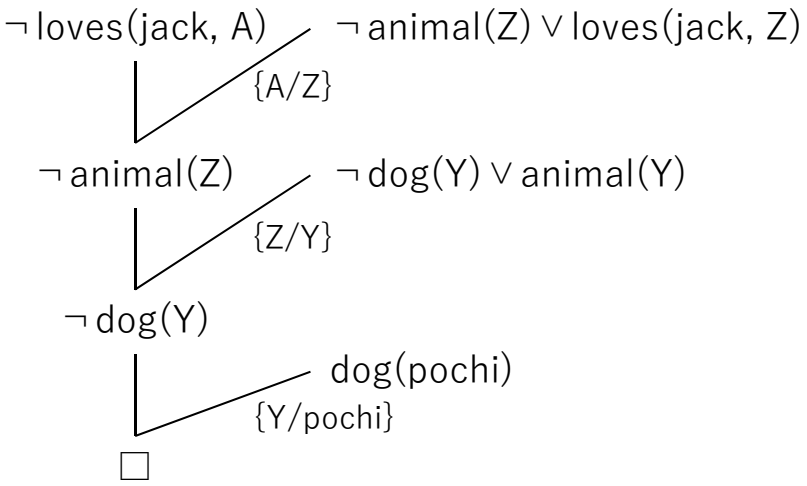
- 文αが変数を含む場合，What型の質問文となる
  - 変数に対する代入が，質問に対する回答に相当する

- 例
  - G = { ¬cat(X) ∨ animal(X)，¬dog(Y) ∨ animals(Y)，¬animal(Z) ∨ loves(jack, Z)，
    cat(mike), dog(pochi) }
  - α = ∃A (loves(jack, A)) : jackが好きなAが存在する．それは何？

  - ¬α = ¬∃A (loves(jack, A)) → ¬loves(jack, A)



{ A/Z, Z/X, X/mike }より，A=mikeが得られる



{ A/Z, Z/Y, Y/pochi }より，A=pochiが得られる

融合法の完全性と健全性

- 述語論理における融合法は，健全である：$\Sigma_0 \vdash_r \alpha \Longrightarrow \Sigma_0 \vDash \alpha$
- 述語論理における融合法は，反駁完全である：$\Sigma_0 \vDash \alpha \Longrightarrow \Sigma_0 \wedge \neg\alpha \vdash_r \square$

- 融合法の反駁完全性の証明の流れ
  - 述語論理の節集合の反駁完全性問題→（エルブランの定理）→
    命題論理の節集合の反駁完全性問題→（基礎完全性定理）→
    反駁証明の構築→（持ち上げ補題）→
    述語論理上の反駁証明

  - 詳しくは参考文献を参照のこと
  - ここでは大まかな流れだけを示す

- エルブラン領域（Herbrand Universe）：考えられる項の集合
  - 節集合に現れるすべての定数と関数記号から生成される基礎項の全体集合
  - 定数が存在しない場合は，適当に一つの定数aを導入する
  - 関数記号がある場合は，無限の集合になる
  - 例：{ nat(0), ¬nat(X)∨nat(s(X)) } のエルブラン領域：{ 0, s(0), s(s(0)), s(s(s(0))), … }
- エルブラン基底（Herbrand base）：考えられる基礎原子式の集合
  - エルブラン領域の要素を引数とする基礎原子式の全体集合
- エルブラン解釈（Herbrand Interpretation）：エルブラン基底に対する解釈
- 基礎節集合：
  - 節集合に対し，全称限量変数にエルブラン領域中の項を代入して得られる文の全体集合
- 補題：節集合Sの基礎節集合Γが，命題論理式の集合として充足可能であれば，Sは充足可能である（Γは変数を含まないので，各原子式を命題記号に置き換えれば，Γは命題論理式集合となる）
- エルブランの定理
  - 任意の節集合Sが充足不能なら，その基礎節集合Γの有限部分集合の中に，充足不能なものが存在する
- 基礎完全性定理：（命題論理における反駁完全性定理そのもの）
  - 基礎節集合Δが充足不能なら，融合規則のみを用いてΔから矛盾を導くことができる
- 持ち上げ補題（lifting lemma）：基礎節集合の反駁証明を，節集合上での反駁証明に対応付ける
  - 基礎代入例同士の融合節は，元の二つの節の融合節に基礎代入を施したものとなる

$$\frac{C_1' \quad C_2'}{C'} \Longleftrightarrow C'\theta = C, C_1'\theta = C_1, C_2'\theta = C_2, \frac{C_1 \quad C_2}{C}$$

融合法の包摂による強化

- 融合法：反駁完全ではあるが，完全ではない
  - 融合法＋包摂は完全
- 定義：包摂（subsumption）
  - 二つの節PとQに対し，条件$P\theta \subseteq Q$を満たす代入が存在するとき，節PはQを包摂（subsume）するといい，$P \succcurlyeq Q$と表す．（各節はリテラルの集合と見做す）
- 例：(1) $p(X) \succcurlyeq p(a)$,　(2) $p(X) \vee \neg q(X) \succcurlyeq p(X) \vee \neg q(X) \vee \neg r(X)$,　(3) $p(X) \vee p(Y) \succcurlyeq p(Z)$
  - (3)の例に注意．節をリテラルの集合と見做す
- Prologによる包摂チェック
  - 節をリストとして与える（各リテラルを","で並べ，全体を[ ] で囲む）

```
%CとDはリストとして与える

theta_subsumes( C, D ):-
   ¥+((¥+((
        copy_term(D, Dcopy),
        numbervars(Dcopy,0,_),
        is_subset(C, Dcopy))))).

is_subset([],_D).
is_subset([H|T],D):-
        member(H,D), is_subset(T,D).
```

```
?- theta_subsumes([p(X)], [p(a)]).
true.

?- theta_subsumes([p(X),-q(X)], [p(X),-q(X),-r(X)]).
true.

?- theta_subsumes([p(X),p(Y)], [p(Z)]).
true.
```

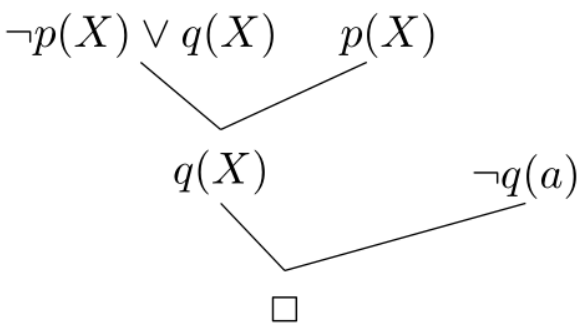
融合法の包摂による強化

- 融合法：反駁完全ではあるが，完全ではない
  - 融合法＋包摂は完全
- 定義：演繹証明手続き
  - Σを節集合，Cを節とする．もしCが恒真か，条件$\Sigma \vdash_r D \wedge D \succcurlyeq C$を満たす$D$が存在するとき，$\Sigma$から$C$が演繹されるといい，$\Sigma \vdash_d C$と表す
- 定理：演繹証明手続きの完全性・健全性
  - Σを節集合，Cを節とする．$\Sigma \models C$が成り立つとき，かつそのときに限り$\Sigma \vdash_d C$が成り立つ

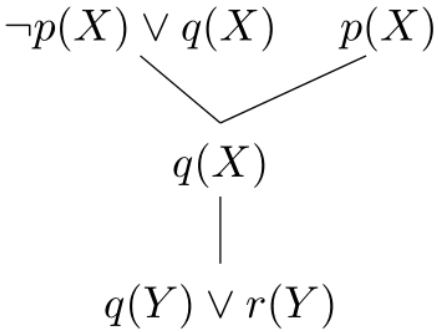
$$\sum = \left\{ \begin{array}{l} q(X) \vee \neg p(X) \\ p(X) \end{array} \right\}$$
$$C = q(Y) \vee r(Y)$$

$$\sum \vdash_r q(Y), \quad q(Y) \succeq q(Y) \vee r(Y)$$
$$\text{より} \sum \vdash_d C \Longleftrightarrow \sum \models C$$

反駁証明の例



演繹証明手続き



融合法でCが導出されたとき，$C\theta \subseteq D$なるDも正しいとして導出するということです．
なぜこれで良いのでしょうか？直観的には…
　Cに含まれる変数は全称限量されている．$C\theta$はCの具体化（具体例）であるので正しいと判断できる
　Dは$C\theta$にリテラルを追加した節 $C\theta \vee \cdots$ であるので，（$C\theta$が正しければ）Dも正しいと判断できる

補足：等号の取り扱い

- 3つの方法
  1. 知識ベースに等号関係に関する文を書き下して，等号を公理化する
  2. 単一化アルゴリズムを拡張する
  3. 推論規則を追加する

- 等号の公理化
  - 等号関係：反射的，対称的，推移的
  - 任意の述語と関数中の等号に対して，等号の代入が可能
  - ∀x (x=x)
  - ∀x, y (x=y ⇒ y=x)
  - ∀x, y, z (x=y ∧ y=z ⇒ x=z)
  - ∀x, y (x=y ⇒ (p(x) ⇔p(y)) # 各述語pに対してそれぞれ必要
  - ∀w, x, y, z (w=y ∧ x=z ⇒ (f(w,x) = f(y,z))) #各関数記号fに対してそれぞれ必要

- 単一化アルゴリズムの拡張
  - 項がある代入の下でほぼ等しい（＝等しいと判断できる）なら単一化可能とする.
  - 例：項 (1 + 2) と項(2 + 1)
    - 本来は，単一化可能ではないが，x+y = y+xを前提に，空代入で単一化可能とする

補足：等号の扱い

推論規則の追加

- デモジュレーション（demodulation）
  - x, y, z を項，xとzの単一化代入をmgu(x,z)=θ，m[z]をzを含むリテラルとする
  - このとき，単位節 x = y と 節$m_1 \vee \cdots \vee m_n[z]$ から，節$m_1 \vee \cdots \vee m_n[SUBST(\theta,y)]$を得る
    - 要は，xと単一化できる部分をyに置き換える
  - 例：f(X) = Y と p(f(A), B) ∨ q(B, C)から，p(Y,B) ∨ q(B,C)を得る
- パラモジュレーション（paramodulation）
  - デモジュレーションの対象は，単位節X=Y．これを X=Yを含む節に拡張したもの
  - 節$l_1 \vee \cdots \vee m_k \vee x=y$ と節$m_1 \vee \cdots \vee m_n[z]$から

    $SUBST(\theta, l_1 \vee \cdots \vee m_k \vee m_1 \vee \cdots \vee m_n[y])$を得る
  - 例：p(X,Y) ∨ f(X)=Y と q(f(A), B) ∨ r(A, B)から，　p(X,Y) ∨ q(Y,B) ∨ r(X,B)を得る

融合法の戦略

融合法の戦略

- 融合法（融合証明手続き）自体は，非決定性を含む
  - 「Σ中から融合可能な節の対を選び，融合を行い融合節Cを得る」
  - 融合可能な節の対が複数あるときに，どの節を選ぶかは明示されていない
  - 選択の方法によって効率（証明に必要となるステップ数）が異なる→効率的な戦略が必要
    - 単位節選好・支持集合戦略・SL融合法など

$$\begin{cases} q \vee \neg a \\ q \vee \neg b \\ b \vee \neg c \\ c \vee \neg a \\ a \end{cases}$$

$$\frac{q \vee \neg a \quad a}{q}$$

$$\frac{\dfrac{b \vee \neg c \quad c \vee \neg a}{b \vee \neg a} \qquad a}{\dfrac{q \vee \neg b \qquad\qquad b}{q}}$$

融合法の戦略

- 前提：融合法による反駁証明 → 論理式に証明したい文の否定を加えて矛盾を導く
- 単位節選好
  - 融合する節の対の内，一方が単位節となっているものを優先する
  - ∵単位節を融合すると，融合節の長さは，元の節よりも短くなる
- 支持集合戦略
  - 支持集合と呼ぶ（論理式集合の）部分集合を考える．
    - 支持集合の初期値は，「証明したい文の否定」
    - 支持集合は「矛盾が含まれるかも知れない文の集合」と考えると分かりやすい？
  - 融合では，支持集合中の文と，他の文を融合する．また，融合結果を支持集合に加える．
    - 気持ちとしては，「証明したい文」から始め，そこから導出された融合節を利用する　ということ
- SL融合法（selective linear resolution）
  - 選択関数を用いて，融合する二つの節 及び 各節の中で消去されるリテラルを決定する
  - 融合される節の一つは，直前の融合節を用いる
- 入力融合（input resolution）
  - SL融合に更に制限を加えたもの
  - 融合には，直前の融合節 と初めから与えられた節集合の要素を用いる
    - 述語論理に対しては不完全だが，確定節集合（Prolog）に対しては完全

確定節（限定子節）と後ろ向き推論・前向き推論

- （命題論理と同様）入力の形を限定すると，より効率的な推論が可能

- 確定節（definite clause）：一つの正リテラルを含む節　　節：リテラルの選言
  - 正リテラルのみから確定節：事実
  - 正負リテラルからなる確定節：ルール $\neg b_1 \vee \cdots \vee \neg b_n \vee h$ → $(b_1, \cdots, b_n) \Rightarrow h$
  - 値域限定条件：hに含まれる変数はb1…bnに含まれる
  - datalog：値域限定条件を満たす，関数記号を含まない確定節
  - ホーン節：確定節＋一貫性制約（正リテラルを持たない節）

- Datalogの後向き推論（backward reasoning）＝入力融合法
  - 「ゴールが成り立つかを調べる」
    - $(b_1, \cdots, b_n) \Rightarrow h$ ：hが成り立つかを調べたい．
    - そのためには，$b_1, \cdots, b_n$が成り立つかを調べればよい．
  - 利点
    - ゴール指向なので，無駄が少ない
    - 縦型探索的で，省メモリ
    - 完全で健全
  - 欠点
    - プログラミング時には，無限ループに注意

確定節（限定子節）と後ろ向き推論・前向き推論

- Datalogの前向き推論（forward reasoning）
  - 「分かっていることから，新たに分かることを導出する」
  - 手順１：Fを事実（基礎原子式）の集合で初期化する
  - 手順２：「前提が充足される節を発火させ，その帰結を集合Fに加える」を繰り返す
    - 知識ベースからわかること＝Fに含まれる事実
  - 利点
    - 単一化がマッチングに簡略化される
    - 不動点が最小モデルと一致する
    - 健全で完全
  - 欠点
    - 証明したいことと無関係なことも導出（計算量が大きい）
    - 値域限定条件を満たさない確定節には適用ができない

ファクト集合が導出される．
複数のことを証明したい場合は，
それらの連言を本体部，
必要な変数を含む特別な述語を頭部に
持つ新たなルールを追加する

例1：p(a)∧q(b)を証明したい場合
p(a), q(b)⇒ans.　を追加

例2：p(X)∧q(X,Y)を証明したい場合
p(X), q(X,Y)⇒ans(X,Y).　を追加

	KB：知識ベース　　$F = \{\}$：導出される知識集合
(1)	$F_{new} = \{f \in KB \mid f$ は事実 $\}$
(2)	**while** $F_{new} \neq \{\}$
(3)	$F = F \cup F_{new}$
(4)	$F_{new} = \{h\theta \mid (b_1, \cdots, b_n \Rightarrow h) \in KB, \{b_1\theta, \cdots, b_n\theta\} \subseteq F, h\theta \notin F\}$
(5)	return F

p(a).　r(b).
p(X)⇒q(X).
q(A),r(B)⇒s(A,B).

F = { p(a), r(b) }
F = F ∪ { q(a) }
F = F ∪ { s(a,b) }

まとめ：今回の授業の内容

- 述語論理の標準形
  - 冠頭標準形
  - スコーレム標準形
  - 節形式

- 述語論理の推論
  - 代入と最汎単一化
  - 融合法
  - 融合法の戦略

用語のまとめ

- 項：オブジェクトの論理表現
  - 関数記号（function symbol）：構造を持った複雑なオブジェクトを表現するための手段
  - 複合項（compound term）：関数記号を伴う項
  - 基礎項（ground term）：変数を全く含まない項
- 文
  - 原子文（atom, atomic sentence）：オブジェクト間の関係，オブジェクトの性質の論理表現
  - 基礎原子文（ground atom）：すべての引数が基礎項である原子文（変数を含まない原子文）
  - 複合文（complex sentence）：論理結合子を用いて原子文を連結した文
- 限量
  - 全称限量（universal quantifier）：すべてのオブジェクトに対する記述
  - 存在限量（existential quantifier）：あるオブジェクトに対する記述
  - 束縛変数（bound variable）：スコープ内にある変数・限量化されている変数
  - 閉論理式（閉じた文）：すべての変数が束縛されている述語文
- 解釈
  - 領域：述語文αが表現する世界での対象の集合
  - 解釈：述語文αの領域をDとする．このとき，αの解釈を以下のステップで与える
    - A中の定数記号，関数記号に，領域中の要素を対応させる
    - 対応関係の下で，基礎原子文への真理値の割り当てる
- 標準形
  - 冠頭標準形：式の左端（先頭）ですべての変数が限量された述語文
  - スコーレ無標準形：冠頭標準形＋「母式が連言標準形」＋「存在限量子を含まない」
  - 節集合：「すべての変数が左端で全称限量された節」の連言