計算の理論レポート（担当：小林直樹先生）
論理プログラミングの基礎

司馬博文 J4-190549

2020 年 7 月 28 日

概要

Prolog での論理プログラミングを基礎付ける数理論理学の事柄について，今後の勉強に活かす目的で数学書的な様式でまとめた．

目次

1 一階述語論理についての用語の準備

文献 [1] で学習した経験を元に，[4] も参照しながらまとめた．

記法 1.1. 言語 L を 1 つ定め，変項を $x,y,z,\cdots$，言語 L に含まれる定数を $a,b,c,\cdots$，述語を $P,Q,R,\cdots$ と書くこととする．また，Prolog での記法に倣い，述語の arity $n\in\mathbb{N}$ を P/n というように述語記号に/を挟んで添えて書く．

L の有限列として一致するという同値関係を $\equiv$ で表す．

定義 1.2 (term)**.** **L-項**全体の集合 Tm_L を次の構成子により帰納的に定める．

1. $x,y,z,\cdots,a,b,c,\cdots\in Tm_L$.
2. $f:Tm_L^n\to Tm_L\quad(f^n\in L)$.

変数 $x,y,z,\cdots$ を含まない L-項を L-閉項 (L-closed term) ／または L-基礎項 (L-ground term) と言う．

定義 1.3 (literal)**.** 次のいずれかの形をした論理式を，**原子論理式 (atomic formulta)** と言う．

$$t_1=t_2,\qquad R(t_1,\cdots,t_n),\qquad (t_i\in Tm_L,R/n\in L).$$

原子論理式とその否定形を **literal** と言う．

定義 1.4 (clause, Horn clause)**.** 次の形をした論理式を，**節**と言う．

$$L_1 \vee L_2 \vee \cdots \vee L_n, \qquad (L_1, \cdots, L_n \text{ are literals}).$$

また，否定を含まない literal（＝原子論理式）を含んだとしてもたった 1 つである節を，**ホーン節**と言う．なお，それぞれの場合について，全てが否定形の literal からなる節を **goal 節**，原子論理式を 1 つのみ含む節を**確定 (definite) 節**または strict Horn clause と言う．

例 1.5. 空な論理式は goal 節である．また，positive(=unnegated) literal 1 つのみからなる節は確定節であるが，これを特に**単位節 (unit)** といい，その中でも特に変数を含まないものを**事実 (fact)** という．

注 1.6. Horn clause を記号 $\rightarrow$ で書き換えると，第一形式を用いて次のようにかける．なお，第一形式 $\rightarrow$ と第二形式 $\leftarrow$ に意味論的差異はない．

$$L_2 \wedge \cdots \wedge L_n \rightarrow L_1 \qquad (\text{確定節の場合}),$$
$$L_1 \wedge L_2 \wedge \cdots \wedge L_n \rightarrow \qquad (\text{ゴール節の場合}).$$

注 1.7 (Prolog では)**.** これら確定節を Prolog の構文で書くと，次のようになる．

Listing 1　Prolog

```
1        L1 :- L2, L3, ..., Ln.
2        L1.
```

また，単位節は，記号:-を省略して 2 のように表される．これは L1 :- true の省略系だと見做せる．ただし，true/0 を L-述語とした．1 の形をした Horn 節を rule，2 の形をした Horn 節を fact と呼ぶ．記号:-に対して，左辺を頭部 (head)，右辺を身体 (body) または目標 (goals) と言う．

定義 1.8 (formula)**. L-論理式**とは，次のように帰納的に定義される記号列のことである．

1. literal は論理式である．
2. 論理式 φ, ψ について，$(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \rightarrow \psi)$ は論理式である．
3. 論理式 φ と変数 x について，$(\exists x\varphi)$, $(\forall x\varphi)$ は論理式である．

L-論理式全体の集合を Fml_L と書くこととする．

以降，一般の L-論理式についてではなく，次の 2 種の標準形のみに議論を絞る．

定義 1.9 (prenex normal form)**.** 次の形の論理式を**冠頭標準形**の論理式という．

$$Q_1 x_1 \cdots Q_n x_n \theta \quad (n \geq 0, Q_i \in \{\exists, \forall\}, \theta \text{には量化記号なし})$$

この時 θ をこの論理式の母式 (matrix) という．

定義 1.10 (Skolem normal form)**.** 冠頭標準形の論理式 φ について，φ 中の各存在量化記号 $\exists y_l$ $(l \in \mathbb{N})$ について，それより前に全称量化記号が n_l 個 $\forall x_1^l, \cdots, \forall x_{n_l}^l$ とあるとすれば，n_l-変数の新たな関数記号 f_l を導入して，φ の母式中の変数 y_l を $f_l(x_1^l, \cdots, x_{n_l}^l)$ で置き換えることで，存在量化記号 $\exists y_l$ を消去することができる．これは拡張された言語 $L \cup \{f_i\}$ での $\forall$-論理式であり，これを **Skolem（連言）標準形**という．

次が成り立つという意味で，これは標準形である．

命題 1.11. φ のモデルに対して，適切に Skolem 関数記号 f_i に解釈を追加することでその Skolem 標準形 φ^S のモデルが得られ，φ^S のモデルから Skolem 関数への解釈を削れば φ のモデルが得られる．

確かに，φ での各存在量化子で縛られた変数 y_l に対して，y_l が存在することと f_l が構成可能であることに等価である．

2 Herbrand の定理

Herbrand の定理とは，Jacques Herbrand が 1930 年に提出した証明論に於ける定理であり [3]，自動定理証明の理論的基盤となった．しかし，直接的に証明を探索する問題は極めて困難で，Prolog の誕生は 1965 年の John Alan Robinson による導出原理の提出を待たねばならない．この節は主に [1],[6] の文献を参照し，基本的には自分の言葉でまとめた．Herbrand の定理と Prolog の処理系が実際に行っていることとの関係が見えにくかったが，系 2.8 の形にまとめればわかりやすいと思われる．

定義 2.1 (Herbrand universe)**.** L-論理式 A の**エルブラン領域**とは，A に出現する定数記号と自由出現する変数記号について，A に現れる関数記号 f から帰納的に生成される<u>L-閉項</u>（＝点）全体のことをいう．ここで，A に現れる記号（定数記号や自由出現する変数記号や関数記号 f など）を $L_A \subset L$ と置く．ただし，A に定数記号も自由変項も存在しない場合は，定数記号を L から自由に 1 つ選んで L_A のただ一つの元とする．

注 2.2. 平たく言えば，Herbrand 領域とは L-式 A に出現する記号からなる集合 L_A を構成子として生成される L_A-閉項 (ground term) 全体の集合のことであると言える．

定義 2.3 (Herbrand normal form)**.** 冠頭標準形の L-論理式 A に対して，その Herbrand 標準形 A^H を，Skolem 標準形の双対として定義する．即ち，次の通りに定義する．

φ 中の各<u>全称</u>量化記号 $\forall y_l$ $(l \in \mathbb{N})$ について，それより前に存在量化記号が n_l 個 $\exists x_1^l, \cdots, \exists x_{n_l}^l$ とあるとすれば，n_l-変数の新たな関数記号 f_l を導入して，φ の母式中の変数 y_l を $f_l(x_1^l, \cdots, x_{n_l}^l)$ で置き換えることで，全称量化記号 $\forall y_l$ を消去することができる．これは拡張された言語 $L \cup \{f_i\}$ での $\exists$-論理式であり，これを **Herbrand（選言）標準形**という．

なお，この時 Herbrand 領域の構成子の集合 L_A も拡張された．これを $L_H (:= L_A \cup \{f_l\}_{i \in I})$ とする．

定義 2.4 (instance」$\exists$-形)**.** A を $\exists$-論理式とする．（即ち，量化記号なしの L-論理式 R について，$A \equiv \exists \vec{x}\, R(\vec{x};\vec{a})$ とする）．この時，式の列 $\vec{t}^i = (t_1^i, \cdots, t_n^i)$ の列 $\{\vec{t}^i \mid i \leq r\}$ について，$\vee\{R(\vec{t}^i;\vec{a}) \mid i \leq r\}$ を A の**例**と言う．

特に，式の列の列 $\{\vec{t}^i \mid i \leq r\}$ が全て Herbrand 領域に入っている時は，**H-例**と言う．

Herbrand 標準形 A^H が $\exists$-論理式であることに注意すれば，より一般に A が冠頭標準形の場合について，H-例は定義できる．

定義 2.5 (instance：冠頭標準形)**.** A を冠頭標準形の L-論理式 $\exists \vec{y}, \forall \vec{x}, R(\vec{y};\vec{x};\vec{a})$ であるとする．変数の列の列 $\{\vec{x}^i \mid i \leq r\}$ と L_A-式の列の列 $\{\vec{t}^i \mid i \leq r\}$ に対して，言語 L_A での量化記号なしの論理式 $\vee\{R(\vec{t}^i;\vec{x}^i;\vec{a}) \mid i \leq r\}$ を，A の **H-例**という．

定理 2.6 (Herbrand's theorem：冠頭標準形について)**.** 冠頭標準形の L-論理式 A について，次の 3 条件は同値である．なお，1 と 3 が同値であるという主張を**エルブランの定理**といい，1 と 2 が同値であるという主張を Gödel の完全性定理という．

1. A は（一階述語論理の証明体系にて）証明可能である：$\vdash A$.
2. A は恒真 (tautology) である：$\vDash A$.
3. A のある H-例が恒真である.

注 2.7. これは，任意の一階述語論理の式は冠頭標準形に変換できるから（命題 1.11），任意の恒真な論理式 A が恒真であることの証明は，有限資源内で確認出来ることの理論的保証となっている.

この定理の証明を自分の言葉でまとめることは叶わなかった.

系 2.8. 冠頭標準形の論理式 F,G について，

$$F = \exists \vec{y},\ \forall \vec{x},\ R(\vec{y};\vec{x};\vec{a}), \qquad\qquad G = \neg F = \forall \vec{y},\ \exists \vec{x},\ \neg R(\vec{y};\vec{x};\vec{a}),$$

とする．次の 4 条件（正味 7 条件）は全て同値である.

1. F は恒偽（充足不能）である：$\vDash \neg F$．即ち，G は恒真である：$\vDash G$.
2. F のある H-例の否定 $\wedge\{R(\vec{t}^i;\vec{x}^i;\vec{a})\}_{i\leq r}$ が恒偽である．即ち，G のある H-例 $\vee\{\neg R(\vec{t}^i;\vec{y}^i;\vec{a})\}_{i\leq r}$ が恒真である.
3. F の否定が証明可能である：$\vdash \neg F$．即ち，G が証明可能である：$\vdash G$.
4. F から導出により空節を導くことができる.

［証明］. 条件 1∼3 にて，「即ち」で結ばれた 2 条件が同値であるのは，$G = \neg F$ から従う.
条件 1∼3 の特に論理式 G についての主張は，Herbrand の定理と Gödel の完全性定理 2.6 から従う.
条件 3 が成り立つとする．即ち，$\vdash \neg F$．この時 F を仮定すると $F \wedge \neg F = \bot$ を導くことができる． □

3 導出原理と単一化

単一化のアルゴリズムはすでに Herbrand の論文に含まれていた [5] が，Robinson が導出原理の形にまとめ，現在の論理プログラミングの基礎となった．文献 [1],[7],[8] を中心にまとめた.

定義 3.1 (導出：命題論理)**.** 次の命題論理での演繹規則を**導出原理**という.

$$\frac{l \vee P \quad \neg l \vee Q}{P \vee Q} \quad (l \text{ is literal})$$

上式は前提となる**親節**，下式を**導出節 (resolvent)** という.

また一般に，2 つの節 C_1, C_2 に対して，$L \in C_1, \neg L \in C_2$ を満たす literal L が存在するならば，次の節 C_R を導く規則を導出という.

$$C_R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\neg L\})$$

注 3.2. 筆者は resolution を「解決」に当たる意味を持つ語として用いられていると思い込んでいたが，どうやら「溶解」「融合」の方である．resolvent は溶液くらいの意味なのかもしれない．例えば SLD resolution の定義 3.10 の式がわかりやすいが，SLD resolution では節の集合から，2 つの節ずつ，ある literal を基点として（例えば文献 [8] ではこの literal を”the literal resolved upon in a clause”と表現するところからも語感が分かる）溶け合うようにして導出規則が適用されていき，最終的に一つの節にまとまる．その証明は線型になる．この過程を「融合」「溶解」になぞらえて名付けられたと思われる.

これを一階述語論理上に拡張するにあたって，このような literal L を見つけるにあたって，適切な同一視をする必要がある．これを pattern matching または単一化 (unification) と呼ぶ．これをまずは，最も一般的な形で定義する．

定義 3.3 (substitution)**.** ある論理式に対して，変数の置換を**代入**という．これを $\sigma : dom(\sigma) \to Tm_L$ と表す．$dom(\sigma)$ は置換が定義される論理式に出現する変数に依存するが，$\sigma(x) = x\ (x \notin dom(\sigma))$ と定めることにより，σ は全変数上に一意に拡張される．すると，Tm_L の構成子に従って，Tm_L 上の写像に帰納的に拡張される．

この代入 $\sigma : Tm_L \to Tm_L$ を，式 $t \in Tm_L$ に作用させることを，式 t 中の変数 $x \in dom(\sigma)$ を一斉に $\sigma(x)$ に置換することとし，これを $t\sigma := \sigma(t)$ と書くこととする．

定義 3.4 (instance)**.** 2 つの代入 σ, τ について，ある代入 θ が存在して関係 $\sigma = \tau\theta$ が成り立つことを，σ は τ の**例**であるという．これは 2 つの変数変換が本質的に同じものであることを意味する．

定義 3.5 (unifier)**.** Tm_L 上の等式の有限集合を $E = \{t_i = s_i \mid i < n\}$ とする．

1. 代入 σ が E の**単一化子 (unifier)** であるとは，$\forall i < n,\ t_i\sigma \equiv s_i\sigma$ が成り立つことをいう．
2. E の単一化子 σ が**最汎単一化子 (MGU: Most General Unifier)** であるとは，どんな E の単一化子 τ も σ の例になっていることをいう．この時，$\sigma = \mathrm{mgu}(E)$ と書くこととする．

与えられた集合 E に対して，これに単一子が存在するかどうかを決定する問題を単一化問題という．

定理 3.6 (unification algorithm)**.** 単一化問題（下記）は決定可能である．

> 与えられた等式集合 E が単一化可能でないならば”NO”と答え，単一化可能ならば”YES”と答え，この時の最汎単一化子 $\mathrm{mgu}(E)$ を出力せよ．

定義 3.7 (導出：一階述語論理)**.** 導出原理は，一階述語論理では，導出木の表記法で，次のように書ける．

$$\frac{p(s) \vee C_1 \quad \neg p(t) \vee C_2 \quad \theta = \mathrm{mgu}(s = t)}{(C_1 \vee C_2)\theta}$$

定義 3.8 (refutation)**. 反駁**とは，節の集合から導出原理のみにより空節を導くことをいう．

注 3.9. この語を用いて，系 2.8 の主張の一部は次のように書き直せる．

1. 節の集合 C が恒偽である．
2. 節の集合 C から反駁が成功する．

従って，query として与えられた述語 P の否定と，プログラムとして与えられた Horn 節の列 $H_1 \wedge \cdots \wedge H_n$ との選言 $\neg P \wedge H_1 \wedge \cdots \wedge H_n$ から反駁することが出来たら，$\neg P, H_1, \cdots, H_n$ は充足不能だとわかる．よって，$H_1, \cdots, H_n$ に矛盾がない限り，$\neg P$ が $H_1, \cdots, H_n$ の下で偽であること，即ち P が $H_1, \cdots, H_n$ の下で真（論理的帰結である：$H_1, \cdots, H_n \vdash P$）であることがわかる．

特に，Prolog では，節の中でも Horn 節に導出の対象を制限する．この場合の導出を SLD 導出という．

定義 3.10 (Selective Linear resolution for Definite clause)**.** query として得たゴール節 $L_1 \wedge \cdots \wedge L_i \wedge \cdots \wedge L_n$ と，規則として定義されたホーン節 $L \leftarrow K_1, \cdots, K_m$ に対する導出を **SLD 導出**という．ただし，L_i と L は，単一化

子 $\theta := \mathrm{mgu}(L_i = L)$ の下で単一化されるとする．このとき，示したいゴール節 $\neg L_1 \vee \cdots \vee \neg L_i \vee \cdots \vee \neg L_n$ から次の goal 節が導かれる．

$$\begin{aligned} & (\neg L_1 \vee \cdots \vee \neg (K_1 \wedge \cdots \wedge K_m) \vee \cdots \vee \neg L_n)\theta \\ \Longleftrightarrow\ & (\neg L_1 \vee \cdots \vee \neg K_1 \vee \cdots \vee \neg K_m \vee \cdots \vee \neg L_n)\theta \end{aligned}$$

注 3.11. SLD resolution とは，Robert Kowalski が導入した導出規則に対して，Maarten van Emden がつけた名前である [8]．Linear とは，導出規則を適用する 2 つの節の内の片方を固定し続ける戦略のことをいい，この証明戦略を使った証明が論理式（節）の有限列 $C_1, C_2, \cdots, C_n$ となることから，また Selective とは，節の中で literal として解決される部位が予め定まっている性質をいう．SLD 導出では，確定節では暗黙のうちに頭部の 1 つに定まっており，ゴール節では任意である．Prolog では，プログラム内に記述された順番で照合するという順番があるのみである．

SLD resolution は初めのゴール節を根とした探索木（導出木）を定める．各節はゴール節のいずれかの肯定形リテラルと単一化可能である場合に対応する．最終的に，葉のうち空節であるものと根を結ぶ道が反駁による証明である．Prolog では，back-tracking アルゴリズム（自動後戻りの深さ優先探索）でこの探索を行い，この証明中で構成された解を返すようになっている．

参考文献

[1] 新井敏康著『数学基礎論』（岩波オンデマンドブックス，2011）

[2] 萩谷昌己著『関数プログラミング』（日本評論社，情報数学セミナー，1998）

[3] J. Herbrand, Recherches sur la théorie de la démonstration. Travaux de la Societe des Sciences et des Lettres de Varsovie, Class III, Sciences Mathematiques et Physiques, 33, pp.33-160, 1930.

[4] https://en.wikipedia.org/wiki/Horn_clause（7/26/2020 確認）

[5] https://ja.wikipedia.org/wiki/Prolog（7/27/2020 確認）

[6] https://ja.wikipedia.org/wiki/エルブランの定理（7/27/2020 確認）

[7] https://en.wikipedia.org/wiki/Resolution_(logic)（7/27/2020 確認）

[8] https://en.wikipedia.org/wiki/SLD_resolution（7/27/2020 確認）