プログラミング言語論

第9回

論理型プログラミング言語

担当：犬塚

今日の講義

論理型プログラミング言語パラダイム

- 構成的プログラミング
  - 構成的数学、直観主義論理
- 論理プログラミング
  - 導出原理、反駁
  - ホーン節、SLD導出
  - 論理プログラム

論理とプログラミング言語の関連

- プログラムの意味論、プログラムの検証
  - プログラムの意味や正しさを数理論理的手法で確かめる。
- 構成的プログラミング
  - プログラムの仕様の満足できることを証明
  - 証明＝プログラム
- 論理プログラミング
  - 論理式によってプログラムを記述するプログラミング言語パラダイム
  - 証明＝計算

構成的論理

- 構成的数学、有限の立場、直観主義論理などと呼ばれる立場の論理。
  - Brouwerの直観主義論理

    排中律：P∨¬P、２重否定の法則：¬¬P⇔Pを認めない。
- 命題の証明のためには、具体的にその存在を組み立てないといけないとする立場。
- 計算機数学は、具体的に計算をするための数学なので、この立場は相性が良い。

直観主義論理

- P∨Q　が真であるには、PかQのどちらが真か明示しないといけない。
- ∃x. P(x)　が真であるには、P(a)が真となるaを明示しないといけない。

例：命題：x^y が有理数となる無理数 x, y が存在する。

証明（直観主義の立場では×）

$x = y = \sqrt{2}$ とする。このとき x^y は有理数か無理数のどちらか。

もし、これが有理数であるなら、これが求めるもの。

もし無理数なら $x=\sqrt{2}^{\sqrt{2}}$、$y=\sqrt{2}$とすると　$x^y=\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^2 = 2$ で、有理数となりこれが求めるもの。

いずれにしろ x とyが無理数で x^y は有理数となるものがある。

構成的論理

- 構成的数学、有限の立場、直観主義論理などと呼ばれる立場の論理。
- 命題の証明のためには、具体的にその存在を組み立てないといけないとする立場。
- 計算機数学は、具体的に計算をするための数学なので、この立場は相性が良い。

構成的プログラミング

- 直感主義論理に基づいて、説明つきのプログラム、あるいは、プログラムの自動生成を行うプログラミングパラダイム。

1. プログラムの仕様を形式的(論理式などで)に与える。
2. その仕様を満足することができることを、直感主義の立場で証明する。
3. その証明がプログラム

プログラムは証明であり、その仕様が既に与えられている。

論理プログラミング

- 論理表現を用いたプログラミング言語パラダイム
- 手続きとしてのプログラム（＝howの知識）ではなく、それが何であるのは（＝whatの知識）でプログラムする。

- プログラムの満たすべき条件を論理式で記述する。
- 満たす対象の存在の自動証明を行う。
  →証明によって計算が行われる。
- 処理系＝自動証明記

論理プログラムの例

fact(1,1).

fact(X,Y)

　← X1 is X–1, fact(X1,Y1),Y is Y1∗X.

father(X,Y)

　← parent(X,Y), male(X).

brother(X,Y)

　←　parent(Z,X), parent(Z,Y), male(X).

論理プログラミングと人工知能

- 論理プログラミングは人工知能と相性がよい。
  - 人工知能プログラミング＝「知識＋汎用エンジン」
  - 論理プログラミング
    ＝「whatの知識（論理式）＋自動証明器」
  - 他のAIプログラミング
    状態空間モデル＋探索エンジン
    制約条件＋プランナー
    データ＋学習エンジン
- 第５世代コンピュータプロジェクト（ICOT)

三段論法（MP）

$$\frac{P \qquad P \rightarrow Q}{Q}$$

$$\frac{P \rightarrow Q \qquad Q \rightarrow R}{P \rightarrow R}$$

$$\frac{\cancel{P} \qquad \neg \cancel{P} \vee Q}{Q}$$

$$\frac{\neg P \vee \cancel{Q} \qquad \neg \cancel{Q} \vee R}{\neg P \vee R}$$

- ＭＰや類似の推論形式は、正負のリテラルを打ち消す形式となっている。

節形式

- リテラル(literal)＝命題記号またはその否定
- 節(clause)＝リテラルを選言(論理和)

  例　P∨Q, ¬P∨Q∨¬R
- 節の連言(論理積)は、即ち、和積標準形である。
- 節をリテラルの集合として扱うことが多い。
- 和積標準形を(論理積を省略して)節の集合で表わす＝節集合。
- どんな論理式も、節集合に変形できる。

例

例　論理式　R∧{P∨¬(Q→R)}を節の連言に直す。

R∧{P∨¬(Q→R)}

＝R∧{P∨¬(¬Q∨R)}

＝R∧{P∨(Q∧¬R)}

＝R∧(P∨Q)∧(P∨¬R)

＝{R, P∨Q, P∨¬R}　　：節集合として扱う。

＝{{R}, {P, Q}, {P, ¬R}}　：節もリテラルの集合。

（命題論理版の）導出原理

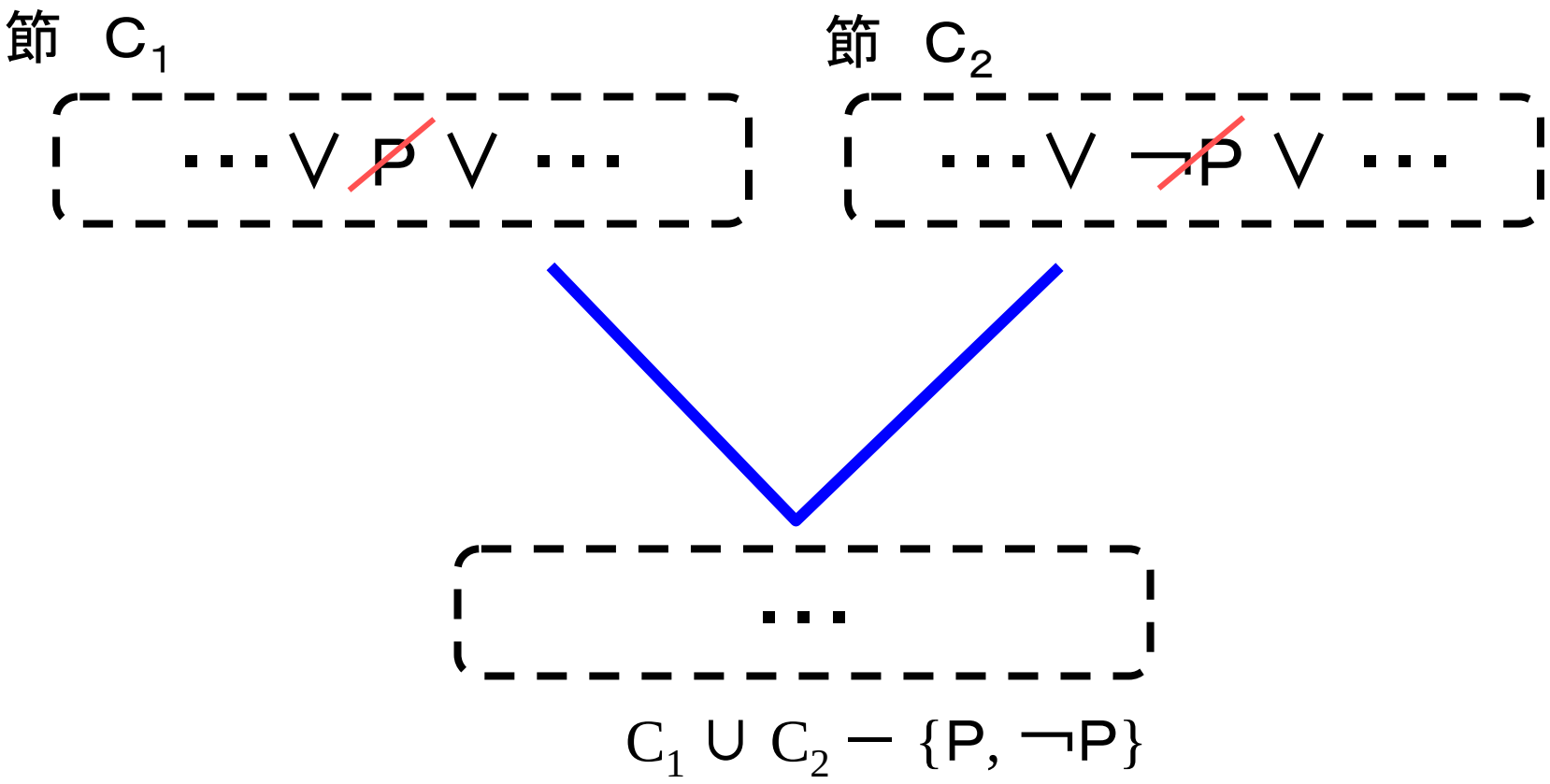
- MPと同様、正負のリテラルを打ち消しあう推論の一般形を導出（融合）（resolution）という。
- 導出は妥当な推論であるという原理を導出（融合）原理(resulution principle)という。

- 導出原理

  2つの節 C_1, C_2と正負のリテラルが $l \in C_1$、$\neg l \in C_2$ に対し、節 $C_1 \cup C_2 - \{l, \neg l\}$は C_1, C_2 から論理的に帰結する。

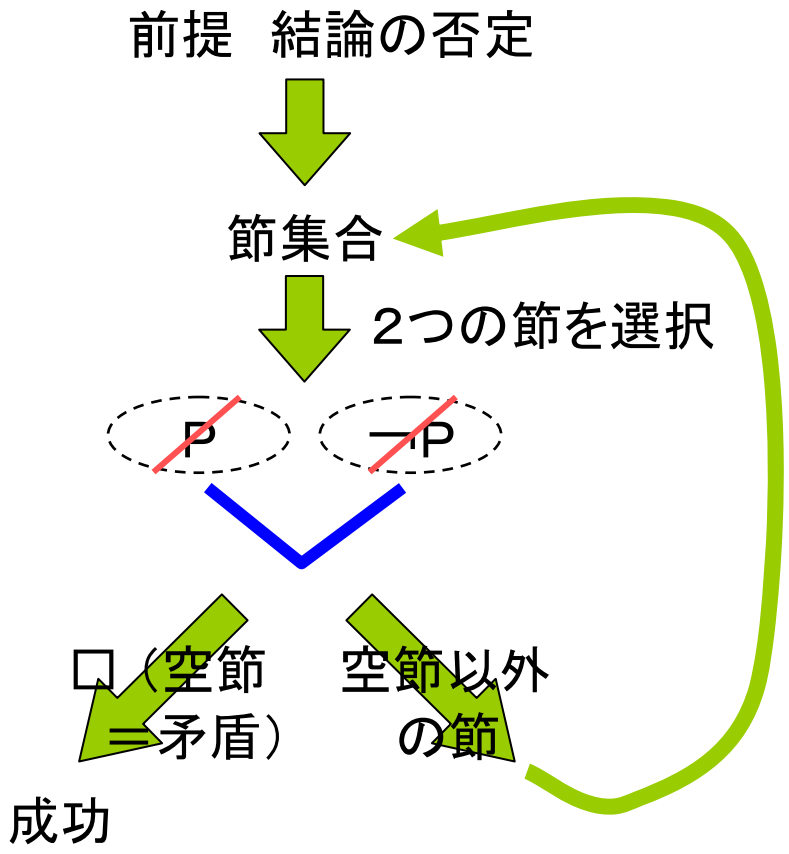
  例　{{R}, {P, Q}, {P, ¬R}}から、Pを得られる。

導出



反駁

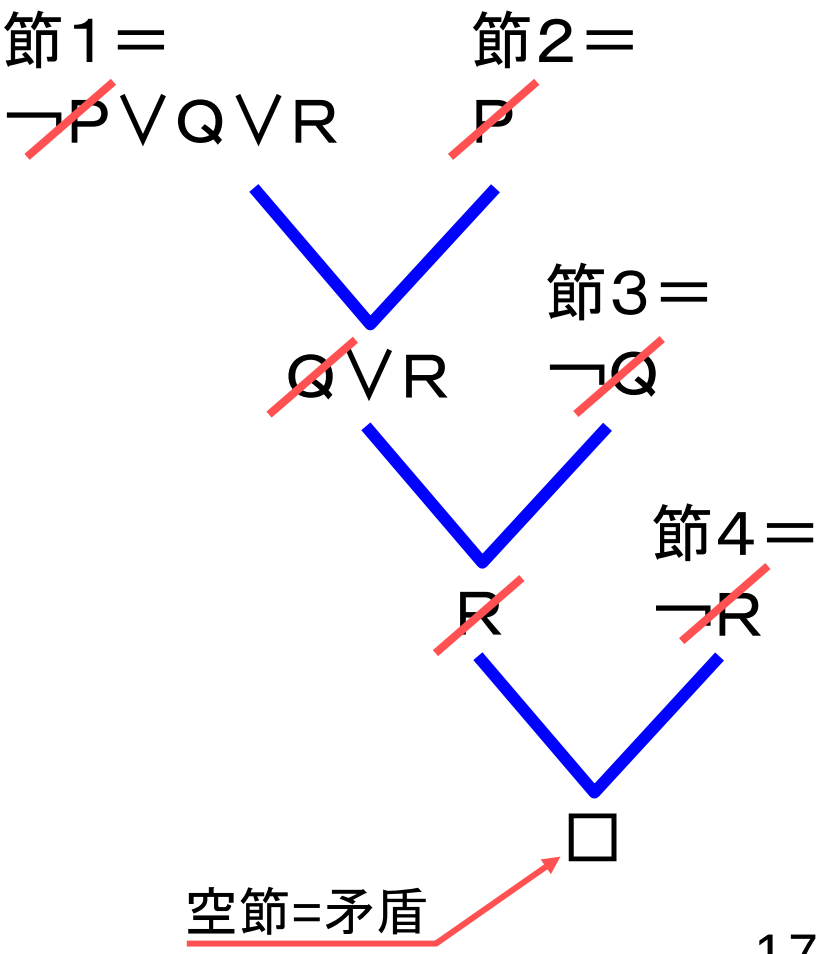
- 導出を用いた背理法証明を反駁(refutation)という。
- 既知の事実(＝前提＝節集合)に、証明したい結論の否定を加え、導出を繰り返す。
- 矛盾が得られることを確認し、結論が正しいことを示す。



導出原理を用いた反駁の例

前提１：くじに当たれば、自転車
　か旅行券をもらえる。
前提２：くじに当たった。
前提３：自転車をもらっていない。
結　論：旅行券をもらった。
Ｐ＝くじに当たった。
Ｑ＝自転車をもらった（もらう）。
Ｒ＝旅行券をもらった（もらう）。

前提１　＝Ｐ→（ＱＶＲ）
　　＝ ¬ＰＶＱＶＲ
前提２　＝Ｐ
前提３　＝¬Ｑ
結　論　＝Ｒ



述語論理における節

- 述語論理のリテラル
  ＝原子論理式（述語記号＋項）か、その否定
- 述語論理の節
  ∀x1…∀xn. (リテラルの選言)
  リテラルに現れる変数記号が始めの部分ですべて束縛。
- すべての節が全称限定されているので、∀を省略。

- どんな述語論理式も恒偽性を保存したまま、節の連言の形式に変形することができる。

（反駁をつかって証明する限りは、恒偽性＝矛盾を導くことができるという性質＝を保証できれば十分）

（述語論理版）導出原理

- 次の2つの原理を組合わせる。
  - 2つの節に正負のリテラルがあるとき、これを打ち消す。

    $l \in C_1$、$\neg l \in C_2$のとき節 $C_1 \cup C_2 - \{l, \neg l\}$を得る。
  - ∀x.P から、P中のxに任意の項を代入した式を得る。
- 変数 $x_1, x_2, \ldots$ を項 $t_1, t_2, \ldots$で置換える代入（置換）を$\theta = \{ t_1/x_1, t_2/x_2, \ldots \}$と書く。

- 導出原理

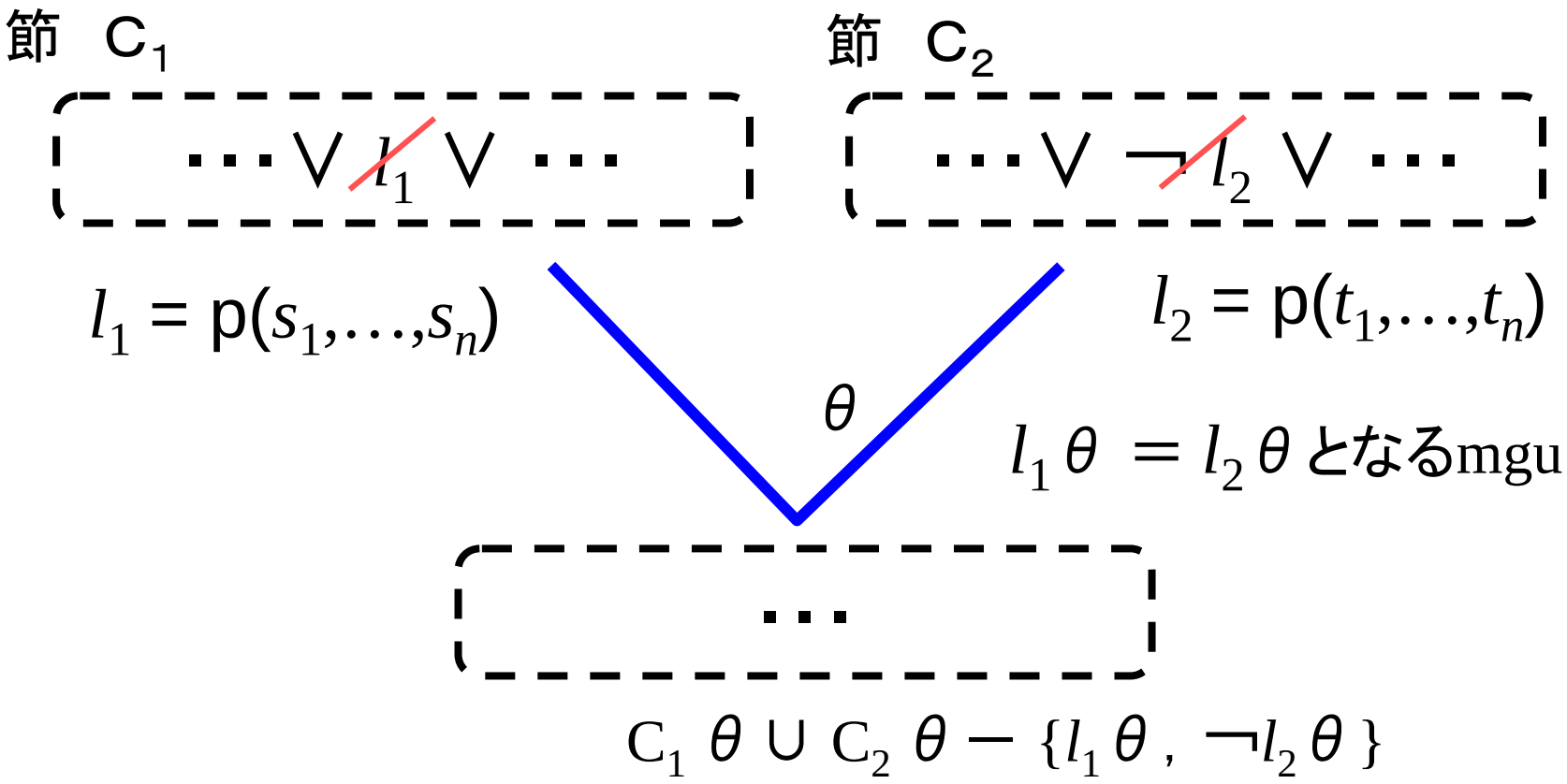
  2つの節 C_1, C_2とリテラルが $l_1 \in C_1$、$\neg l_2 \in C_2$に対し、置換θが$l_1\theta = l_2\theta$とするとき、

  節 $C_1\theta \cup C_2\theta - \{l_1\theta, \neg l_2\theta\}$は C_1, C_2 から論理的に帰結する。

単一化、最汎単一化

- 変数 $x_1, x_2, \ldots$ を項 $t_1, t_2, \ldots$で置換える代入（置換）を $\theta = \{ t_1/x_1, t_2/x_2, \ldots \}$と書く。
- 2つのリテラルが $l_1 \in C_1$、$\neg l_2 \in C_2$に対し、$l_1\theta = l_2\theta$となる代入（置換）を、単一化置換（単一化子；ユニファイア）という。
- $l_1\theta = (l_1\rho)\sigma$のように、$\theta$を2つの置換の合成で表せるとき、$\rho$は$\theta$より一般的であるという。
- 最も一般的な単一化子を、最汎単一化子（mgu; most general unifier）という。

（述語論理版）導出


節 C_1
$\cdots \vee l_1 \vee \cdots$
$l_1 = p(s_1,\ldots,s_n)$
節 C_2
$\cdots \vee \neg l_2 \vee \cdots$
$l_2 = p(t_1,\ldots,t_n)$
θ
$l_1\theta = l_2\theta$ となるmgu
$\cdots$
$C_1\theta \cup C_2\theta - \{l_1\theta, \neg l_2\theta\}$

述語論理版の反駁

- 論理式を節集合（全称限定された節集合）で表す。
- 節集合から、2つの節を取り出す。
- この2つに導出原理を適用する。
  - 各節から、1つづつのリテラルに注目する。
  - そのリテラルにmguがないか探し、あればそれを用いて導出を適用する。
- 空節が導かれれば成功、そうでなければ導出を繰り返す。

節形式（clausal form）

節中の正のリテラル／負のリテラルに分けて書くと次の通り。

$(P_1 \vee \cdots \vee P_n) \vee (\neg Q_1 \vee \cdots \vee \neg Q_m)$

これは、次の通り変形できる。

$(P_1 \vee \cdots \vee P_n) \vee \neg (Q_1 \wedge \cdots \wedge Q_m)$

$= P_1 \vee \cdots \vee P_n \leftarrow Q_1 \wedge \cdots \wedge Q_m$

これをさらに省略して次のように書く。

$P_1, \cdots, P_n \leftarrow Q_1, \cdots, Q_m$

この形式を節形式（clausal form）という。

節形式では、←の左辺の「,」は選言、右辺の「,」は連言。

ホーン節

節形式、

$P_1, \cdots, P_n \leftarrow Q_1, \cdots, Q_m$

において n≦1のとき、ホーン節（Horn clause）という。

つまりホーン節は次の形式。

- $P \leftarrow Q_1, \cdots, Q_m$
- $P \leftarrow$
- $\leftarrow Q_1, \cdots, Q_m$
- $\leftarrow$

ホーン節

ホーン節は次の4形式に分けられる。

- $P \leftarrow Q_1, \cdots, Q_m$

「・・・ならば・・・である」という形。規則（ルール）節という。

- $P \leftarrow$

無条件に「・・・である」という形。事実節という。

- $\leftarrow Q_1, \cdots, Q_m$

「・・・ならば矛盾である」、「・・・ということはない」という形。
結論の否定から矛盾を導く（反駁）で用いる。
「・・・ですか？」と質問しているのと同じ。ゴール節という。

- $\leftarrow$

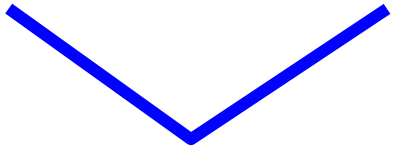
「矛盾」を表す。矛盾節、空節という。

SLD導出

- プログラミング言語Prologでは、ホーン節に限定して、反駁用いる。
- さらに、導出を次のとおり限定：
  - 導出の片親を、結論の否定＝ゴール節とする。
  - この導出によって得られた節次のゴール節とする。
  - 導出に用いるリテラルを、ゴール節の最左リテラルとする。
  - 導出の結果、もう片親から引き継ぐリテラルを、左側に挿入する。

SLD導出

ゴール節 ←A, ○, …,○.　　A’←△,…, △.



新たなゴール節 ←△,…, △, ○, …,○.

- ゴール節は、プログラムの呼出しと見なせる。
- ゴール節の各リテラルは、副プログラムの呼び出し。
- 他のルール節は、副プログラムそのもの。

- SLD導出は、手続き的なプログラム実行と類似。

まとめ

- 論理に基づいた2つのプログラミングパラダイム。
  - 構成的プログラミング
    - 構成的数学に基づいたプログラミングの概念。

      （研究レベルであって、実用にはなっていない）
  - 論理プログラミング
    - 導出原理と反駁に基づいたプログラミング。
    - Prologの原理。
    - 手続き的な解釈も可能であり、実用的なプログラミングも可能。