知能と複雑系 113－1
(1998. 7. 23)

SLD導出の難点とその克服

赤間 清　　川口 雄一　　宮本 衛市

北海道大学 大学院 システム情報工学専攻

札幌市北区北 13 条西 8 丁目. Tel. 011-706-6814

{akama,yuuichi,miya}@complex.eng.hokudai.ac.jp

SLD レゾリューションの理論は Prolog の理論的基礎を与えている．しかし SLD レゾリューションは 論理的な問題を解くための最適な方法とは いえない．たとえば，SLD レゾリューションは 計算の制御に 不必要な制限を課している．その制限のために，計算の制御は コストが高くなり，コストを下げようとすると，Prolog に見られるように，計算の 柔軟性が損なわれる．本論文では 等価変換パラダイムと 呼ばれる新しい計算パラダイムを採用して，SLD レゾリューションの制限による困難を克服する．

論理プログラミング　　等価変換　　アンフォールド変換　　SLD レゾリューション

Overcoming the Difficulties of SLD Resolution

Kiyoshi Akama　　Yuuichi Kawaguchi　　Eiichi Miyamoto

Division of System and Information Engineering, Hokkaido University

North 13 West 8 Kita-ku Sapporo. Tel. 011-706-6814

{akama,yuuichi,miya}@complex.eng.hokudai.ac.jp

The theory of SLD resolution provides Prolog with a theoretical foundation. SLD resolution is, however, not the best way to solve logical problems. For instance, SLD resolution imposes some unnecessary restrictions on the control of computation. These restrictions make computation control more expensive and, conversely, adoption of low-cost control would make computation less flexible, as can be seen in the Prolog case. This paper adopts a new computation paradigm, called Equivalent Transformation paradigm, and overcomes the difficulties due to restrictions of SLD resolution.

logic programming, equivalent transformation, unfolding, SLD resolution

1 まえがき

論理プログラミングは，問題を論理式で記述する宣言的側面（what）と，推論により論理式を変形する手続き的側面（how）を持つ．SLD 導出の理論 (健全性と完全性) は，両者を結合している．すなわち，論理式で記述された「求めるべき解」は，SLD 導出によって 過不足なく求めることができる．論理パラダイムは，制約や継承などを扱うように，さまざまに拡張されてきたが，その多くが，計算を推論とみなす計算モデルを基礎として，SLD 導出などと類似の計算方式を採用している．

しかし，SLD 導出の理論は十分洗練された構造をしているとはいえず，SLD 導出における計算の制御を記述する「計算規則」は，本来備わっているはずの計算の自由度を制限する結果になっている （2章）．

本論文では，SLD 導出の問題点を克服するために，推論を基礎とした方法ではなく，等価変換による解法 [1, 2] を採用する．SLD 導出による解法のクラスを等価変換による解法のクラスの中に埋め込むことができることを示し，SLD 導出の「計算規則」の限界を明らかにする．また，SLD 導出ではなく 等価変換 を基礎とすれば，SLD 導出の 計算規則がもたらす 無用な制限を避けることができることを示す．これは，より簡潔な理論のもとで，より効率的な計算を達成する可能性を示唆する．

本論文の 2章では，「推論による計算」を基礎付ける SLD 導出の理論の限界を指摘し，本論文の議論の方針を述べる．3章では，論理的問題の定義と，それを等価変換で解く方法を復習する．4章では，等価変換の 1 つとして アンフォールド変換を導入し，それを用いて論理的問題を解く方法を議論する．5章では，SLD 導出を用いて論理的問題を解く方法を与える．ここでの方法は，従来の方法を少しだけ変形した方法である．6章では，SLD 導出による解法は，制限されたアンフォールド変換による解法と等価であることを示す．7章では，SLD 導出による解法の問題点を，等価変換による解法に移行することによって克服する可能性について述べる．

2 問題の提起

2.1 定理証明から論理プログラミングへ

論理プログラミングは，定理証明の研究から生まれた．確定節 P と アトム q が与えられたとする．P は プログラム，q は 問い合わせと呼ばれる．P から q を証明するために，P と $\leftarrow q$ （q の否定にあたる節）を仮定し，推論によって矛盾を導く．問い合わせ q が変数を含む場合，論理プログラミングでは，変数 に何を代入するとプログラム P の論理的帰結となるかを答えなければならない．これは単なる証明とは違うが，同じく P と $\leftarrow q$ を仮定し矛盾を導く背理法が使える．変数への適切な代入は，背理法による証明の過程から抽出することができる．

2.2 SLD 導出の理論の問題点

SLD 導出の理論は，上記の直観的枠組を理論的に定式化したものである．SLD 導出の正当性は，健全性と完全性の定理の形で示されている．健全性は，SLD 導出で得られる解が 常に正しいことであり，完全性は，正しい解は すべて SLD 導出で得られることである．

SLD 導出の枠組は，理論的に整備され，一見何の問題もないように 思えるかもしれない．しかしそれを基にして 推論システムを構築しようとすると SLD 導出の理論の限界が 現れる．たとえば，

1. SLD 導出の理論に基づいて厳密に正当性を保証できる推論システムを構築しようとすれば，計算規則を実現するために多くのコスト（時間的にも空間的にも）がかかる恐れがある．なぜなら 計算規則は写像であり，それを記憶し，計算するコストは一般には 少なくないからである．

2. 計算規則を実現するためのコストを低減する 1 つの方法は，たとえば Prolog のように 単純な計算規則を用いることである．Prolog では 最も左のアトムを選ぶ計算規則を用いて，コストを最低レベルに抑えている．しかしその結果，計算の順序の柔軟性が損なわれ，探索空間が非常に増大する場合がある[1]．

2.3 等価変換による解法への埋め込み

SLD 導出は，論理的な問題を解くための，良く知られた解法のクラスである．それは，計算規則というパラメータを持っており，同じ論理的問題でも，計算規則の如何によって，問題が解けたり，解けなかったりする．解けないのは，たとえば，無限ループに陥って，計算が停止しない場合である．

一方，論理的問題を等価変換によって解くことができることが すでに証明されている [2]．等価変換もまた，論理的問題を解く解法のクラスであり，「各時点で，どんな等価変換ルールを使うか」をそのパラメータと考えることができる．そのパラメータの如何によって，計算効率は大きく変化する．

本論文では，SLD 導出による解法のクラスを等価変換による解法のクラスに「埋め込む」ことを考える．す

[1] そのため Prolog では 多くの制約充足問題が事実上解けない．

なわち，等価変換による解法が，SLD 導出による従来の解法を真に包含していること証明する．そして，2.2 章で述べた SLD 導出の問題点 (計算規則の実現のコスト，計算順序の固定化) が，等価変換による解法のクラスへ移行することによって，解消できることを示す．

本論文では，この議論をさらに進めて，解法のクラスを拡大することによって，よりよい問題解決の枠組を得ることの重要性についても議論する．

紙面の都合上，証明は すべて割愛する．

3 等価変換による論理的問題の解法

3.1 論理プログラム

述語論理のためのアルファベット Δ が与えられているとする．$\mathcal{A}$ をΔ 上のアトム全体の集合，$\mathcal{G}$ をΔ 上の基礎アトム（変数を含まないアトム）全体の集合，$\mathcal{S}$ を Δ 上の代入全体の集合とする．アトムに代入を作用させて得られるアトムを，そのアトムの例と呼ぶ．アトムの基礎例とは，アトムの例で，しかも基礎アトムであるものをいう．アトム a の基礎例全体の集合を $rep(a)$ で表す．アトム a の述語が r であるとき，a を r アトムともいう．

Δ 上の確定節とは，$\mathcal{A}$の元 $H, B_1, \cdots, B_n$から作られた $H \leftarrow B_1, \cdots, B_n$ $(n \geq 0)$ の形の式である．H を確定節の ヘッド，$B_1, \cdots, B_n$ を ボディ と呼ぶ．確定節のボディに出現するアトムを ボディアトムと呼ぶ．$\mathcal{G}$ の元だけからなる確定節を基礎節と呼ぶ．節 C の ヘッドを $head(C)$，節 C の ボディアトム全体の集合を $body(C)$ で表す．節 C の ヘッドの述語が r であるとき，C を r 節ともいう．

Δ 上の論理プログラム （logic program）とは，Δ上の確定節の集合である．

3.2 論理的問題

Δ 上の論理プログラム P と，問い合わせ $q \in \mathcal{A}$ の 2 項組 $\langle P, q\rangle$ を論理的問題と呼ぶ [2]．これは，$\mathcal{G}$ の 部分集合

$$L(P,q) = \{g \mid P \models (g \leftarrow), g \in rep(q)\}$$

を求める問題である[2]．$L(P,q)$ を論理的問題 $\langle P, q\rangle$ の解集合と呼ぶ．

3.3 等価変換による論理的問題の解法

等価変換を繰返し適用して得られる 節集合の列

$$P_1 \rightarrow P_2 \rightarrow \cdots$$

を等価変換列と呼ぶ．等価変換列を作ることによって論理的問題 $\langle P, q\rangle$ の解集合 $L(P,q)$ を計算することができる [2]．

[等価変換による解法]

1. q の述語を r とする．Δ の述語全体の集合 R に属さない述語 r'を選ぶ．r アトムが与えられて，述語を r から r'に変更する写像を ϕ とする．
2. 確定節 $(\phi(q) \leftarrow q)$ を G_1 とする．
3. 節集合 $P' = P \cup \{G_1\}$ を，繰返し等価変換して，節集合 $P'' = P \cup F$ を得る．ただし F は r' 節の単位節集合である．r' 節が すべて単位節にならなければ失敗である．このとき等価変換列
   $$P \cup \{G_1\} \rightarrow \cdots \rightarrow P \cup F$$
   が得られる．
4. 論理的問題 $\langle P, q\rangle$ の解集合 $L(P,q)$ を，
   $$L(P,q) = \bigcup_{(\phi(a)\leftarrow)\in F} rep(a)$$
   で求める．これは，上記の等価変換列によって得られる単位節集合 F の中の節 $(\phi(a) \leftarrow)$ から得られる a の基礎例の集合 $rep(a)$ の すべての和集合である．

4 アンフォールド変換による解法

4.1 ユニファイアと最汎ユニファイア

定義 1 x と y を $\mathcal{A}$ の アトムとする．$x\theta = y\theta$ を満たす代入θ を x と y の ユニファイアという． □

定義 2 x と y を $\mathcal{A}$ の アトムとする．次の条件を満たす代入Θ を x と y の 最汎ユニファイアという．

(1) Θ は x と y の ユニファイアである．

(2) x と y の 任意のユニファイア θ に対して，$\theta = \Theta\sigma$ を満たす代入 σ が存在する □

4.2 レゾルベントの定義

レゾルベントの定義を与える．

定義 3 節 C_a,C_bを，

$$C_a = (H \leftarrow A_1, \cdots, A_i, \cdots, A_m)$$
$$C_b = (K \leftarrow B_1, \cdots, B_n)$$

とする．ρ を C_a と $C_b\rho$ が変数を共有しないように選ばれた変数名変更代入 (*renaming*) とする．また θ を A_i と $K\rho$ の最汎ユニファイアとする．そのとき，節 C_a から，そのボディアトム A_i，節 C_b，変数名変更代入 ρ，最汎ユニファイア θ によって得られるレゾルベント (*resolvent*) とは，

[2] $\models$ の定義は，紙面の都合上割愛する．詳細は [2].

$$C_r = (H\theta \leftarrow A_1\theta, \cdots, A_{i-1}\theta, B_1\rho\theta, \cdots, B_n\rho\theta, A_{i+1}\theta, \cdots, A_m\theta)$$

で与えられる節 C_r のことである．これを，

$resolvent(C_a, A_i, \theta, C_b, \rho)$

で表記する． □

以下では，節 C_a から，あるボディアトム A_i，ある節 C_b，ある変数名変更代入 ρ，ある最汎ユニファイア θ によってレゾルベント C_cが 得られるとき，「C_a を展開して C_c を得る」とも言う．

4.3 アンフォールド変換の定義

論理プログラムに対するアンフォールド変換を定義する．

定義 4 論理プログラム $P_1 = \{C_j \mid j \in J\}$ から節 C_a とその ボディ アトム A_i を選んで，変数名変更代入 ρ，最汎ユニファイア θ を用いてアンフォールド変換して P_2 を得る (簡単に，「P_1 を展開して P_2 を得る」とも言う) とは，次のように定義される．

1. $P_1 = \{C_j \mid j \in J\}$ から節
   $C_a = (H \leftarrow A_1, \cdots, A_i, \cdots, A_m)$
   を選ぶ．
2. 節 C_a のボディから A_i を選ぶ．
3. プログラム P_1 の各節 $C_j (j \in J)$ に対して，ρ_j を C_a と $C_j\rho_j$ が変数を共有しないように選ばれた変数名変更代入とする．$K = head(C_j)$ とする．A_i と $K\rho_j$ が単一化可能な j の集合を J' とし，任意の $j \in J'$ に対して，θ_j を A_i と $K\rho_j$ の最汎ユニファイアとする．
4. ρ_j $(j \in J')$ と θ_j $(j \in J')$ を用いて，

$$R = \{resolvent(C_a, A_i, \theta_j, C_j, \rho_j) \mid j \in J'\}$$
$$P_2 = (P_1 - \{C_a\}) \cup R$$

として P_2 を得る． □

よく知られているように，アンフォールド変換は宣言的意味[3] を保存する [6].

定理 1 論理プログラム P_1 からアンフォールド変換して P_2 が得られるとき，$\mathcal{M}(P_1) = \mathcal{M}(P_2)$ が成り立つ． □

[3] 宣言的意味の定義は，紙面の都合上割愛する．詳細は [2].

4.4 アンフォールド変換列による解法

アンフォールド変換を繰返し適用して得られる 節集合の列

$P_1 \rightarrow P_2 \rightarrow \cdots$

をアンフォールド変換列と呼ぶ．アンフォールド変換は宣言的意味を保存するので，論理的問題 $\langle P, q\rangle$ の解集合 $L(P, q)$ をアンフォールド変換列を用いて計算することが考えられる．

[アンフォールド変換による解法] 等価変換による解法において，等価変換の方法を，節集合から r' 節だけを選んでアンフォールド変換するように限定する．

アンフォールド変換を用いた問題の解法では，各回ごとに次のような自由度がある．

1. どの r' 節を選択するか．
2. その節のどのボディアトムを選ぶか．
3. どの変数名変更代入を選ぶか．
4. どの最汎ユニファイアを選ぶか．

アンフォールド変換を用いた問題の解法では，これらを 1 組だけ選んで，新しい節集合を得ることを繰り返す．各回ごとにどの 1 組でも変換は等価になり，2 組以上を選ぶ必要はない．

5 SLD 導出による解法

5.1 SLD 演繹列

本節では SLD 演繹列の定義を与える．ただし，本節の定義は 従来の定義と少し異なる部分がある．それは，従来の定義では “ゴール” が負節（確定節からヘッドを取り除いた節）であるが，本論文では “ゴール” は確定節とした点である．そのヘッドを取り除けば，以下の定義は 従来の定義と同じになる．

新しい定義[4] を採用したのは，伝統的な視点から新しい視点へと移行する準備のためである．5.4節，5.5節 と進むにつれて，新しい定義の有効性が明らかになるであろう．

はじめに 計算規則を定義しておく．

定義 5 単位節でない確定節から，そのボディアトムを 1 つ決定する写像を 計算規則と呼ぶ． □

定義 6 P を Δ 上の 論理プログラム，q をΔ 上のアトム，r を q の述語，r' を，Δ の述語全体の集合 R に属さない述語，ϕ を，r' アトムが与えられて，その述語を

[4] その妥当性は本章の命題や定理が成り立つことで示される．

r から r'に変えた新しいアトムを与える写像，G_1 を 確定節 $(\phi(q) \leftarrow q)$，sel を計算規則とする．

sel による Pに対する G_1 の SLD演繹列 *(derivation)* とは，

(1) 確定節の列:
$$G_1, G_2, G_3, \cdots$$

(2) アトムの列:
$$A_1, A_2, \cdots$$

(3) P の中の確定節の列:
$$C_1, C_2, \cdots$$

(4) 変数名変更代入の列:
$$\rho_1, \rho_2, \cdots$$

(5) 最汎ユニファイアの列:
$$\theta_1, \theta_2, \cdots$$

の５つ組で，各 G_{i+1} $(i \geq 1)$ は，

1. G_i （単位節ではない確定節）から sel によって指定されるボディアトム A_i
2. 節 C_i,
3. 変数名変更代入 ρ_i,
4. 最汎ユニファイア θ_i

を使って得られるレゾルベント
$$resolvent(G_i, A_i, \theta_i, C_i, \rho_i)$$
である．G_1，G_2，G_3，$\cdots$ をゴール節，$C_1, C_2, \cdots$ を入力節という．G_1 を 初期ゴール節という． □

定義 7 計算規則 sel に基づいて，ゴール節を展開する観点から，ゴール節を３つに分類する．

1. sel の指定するアトムを選んで，少なくとも１つの展開が可能なゴール節を未完節という．
2. 単位節であり，したがって展開できないゴール節を成功節という．
3. 単位節ではないが sel の指定するアトムに単一化するヘッドを持つ節が存在せず，展開ができないゴール節を失敗節という． □

定義 8 SLD演繹列を４つに分類する．

1. 無限に続く SLD演繹列を無限列と呼ぶ．
2. 有限の長さの SLD演繹列で，最後のゴール節が成功節であるものを成功列という[5]．
3. 有限の長さの SLD演繹列で，最後のゴール節が失敗節であるものを失敗列という．
4. 有限の長さの SLD演繹列で，最後のゴール節が未完節であるものを未完列という．

無限列でない演驛列を有限列という．有限列の最後のゴール節を最終ゴール節という． □

SLD 演繹列，SLD 成功列，失敗節などがどの計算規則を用いたかを陽に示したいときは，計算規則名を冠して sel-SLD 演繹列，sel-SLD 成功列，sel-失敗節などということがある．

5.2 計算解代入

SLD 導出を使って，論理的問題 $\langle P, q \rangle$ の解集合 $L(P, q)$ を計算することを考える．そのために，P に対する $G_1 = (\phi(q) \leftarrow q)$ の sel-SLD 成功列を すべて求める．sel-SLD 成功列における最汎ユニファイアの列が，
$$\theta_1, \theta_2, \cdots, \theta_{n-1}$$
であるとき，それらの合成
$$\theta_1\theta_2 \cdots \theta_{n-1}$$
を q の変数に制限して得られる代入を計算解代入と呼ぶことにする．

従来の理論では，SLD 導出の目的は 計算解代入を得ることと見なして，理論を構築している．しかし これは いくつかの問題がある．たとえば，

1. それでは 健全性と完全性の定理をすっきり捉えられない．これは 5.3節 で解決する．
2. 計算解代入の概念は 繁雑であり，拡張性に乏しい[6]．これは 5.4節 で解決する．

次の３つの節では 定理 5を証明するために，従来の理論を新しい視点から捉え，段階的に 定理 5の証明に至る議論を行なう．

5.3 代入を基礎とする定理

本節では，SLD 導出の目的は，プログラム P と 問い合わせ q からある条件を満たす代入をすべて得ることであると捉える．これは，伝統的な理論が，SLD 導出は すべての計算解代入を求めるものだと捉えているのを少しだけ修正したものである．すなわち，本節では，次のような問題（問題 A）を解くという目的を考える．

[5] これは 従来の定義の SLD 反駁列 （refutation）に対応する．

[6] 一般的なデータ構造のもとで統一的な議論を展開する場合には，変数の概念や代入の制限の概念は特殊すぎて，一般論をうまく構成できない．

[問題 A]
P をプログラム，q をアトムとするとき，
$P \models (q\sigma \leftarrow)$
を満たす 代入 σ の全体の集合
$satSubst(P,q)$
を求めよ．

q の述語を r とする．Δ の述語全体の集合 R に属さない述語 r'を選ぶ．ϕ は，r アトムが与えられて，その述語を r から r'に変えた新しいアトムを得る写像，G_1 は 確定節 $(\phi(q) \leftarrow q)$，sel は 計算規則とする．

P に対する $G_1 = (\phi(q) \leftarrow q)$ の sel-SLD 成功列から得られる計算解代入 θ と任意の代入 ρ の合成によって得られる代入$\sigma = \theta\rho$ をすべて集めた集合を $sldSubst(P,q)$ とする．

このとき，伝統的な論理プログラミングの理論 [5] の健全性の定理と完全性の定理をあわせて，次の形に言い替えることができる．

定理 2 $satSubst(P,q) = sldSubst(P,q)$ □

この定理は，問題 A を解くためには，$sldSubst(P,q)$ を求めればよいことを示している．

5.4 アトムを基礎とする定理

前節では，SLD 導出の目的は，プログラム P と 問い合わせ q からある条件を満たす代入をすべて得ることであると捉えた．しかし本節では，SLD 導出の目的は，プログラム P と 問い合わせ q からある条件を満たすアトムをすべて得ることであると捉える．

[問題 B]
P をプログラム，q をアトムとするとき，
$P \models (b \leftarrow)$
を満たす，q の例 b 全体の集合
$satAtom(P,q)$
を求めよ．

このとき，前節の定理から容易に次の定理が導かれる．ϕ，G_1，sel などの定義は 問題 A の場合と同様である．P に対する節 $G_1 = (\phi(q) \leftarrow q)$ の sel-SLD 成功列における最終ゴール節 $G_n = (\phi(a) \leftarrow)$ から得られるアトム a のすべての例の集合を $sldAtom(P,q)$ とする．

命題 1 次の 2 式が成り立つ．
$satAtom(P,q) = \{q\sigma \mid \sigma \in satSubst(P,q)\}$
$sldAtom(P,q) = \{q\sigma \mid \sigma \in sldSubst(P,q)\}$

定理 3 $satAtom(P,q) = sldAtom(P,q)$ □

この定理は，問題 B を解くためには，$sldAtom(P,q)$ を求めればよいことを示している．

5.5 SLD 導出による論理的問題の解法

問題をさらに 基礎アトムを求める問題に変化させる．

[問題 C]
P をプログラム，q をアトムとするとき，
$P \models (b \leftarrow)$
を満たす，q の基礎例 g 全体の集合
$satGatom(P,q)$
を求めよ．

このとき，前節の定理から容易に次の定理が導かれる．ϕ，G_1，sel などの定義は 以前と同様である．

P に対する節 $G_1 = (\phi(q) \leftarrow q)$ の sel-SLD 成功列における最終ゴール節 $G_n = (\phi(a) \leftarrow)$ から得られるアトム a のすべての基礎例の集合を $sldGatom(P,q)$ とする．

命題 2 次の 2 式が成り立つ．
$satGatom(P,q) = satAtom(P,q) \cap \mathcal{G}$
$sldGatom(P,q) = sldAtom(P,q) \cap \mathcal{G}$

定理 4 $satGatom(P,q) = sldGatom(P,q)$ □

問題 C を解くためには $sldGatom(P,q)$ を求めればよいことを，この定理は 示している．

5.6 SLD 導出による解法

実は，$satGatom(P,q)$ は論理的問題 $\langle P,q\rangle$ の解集合 $L(P,q)$ に外ならないので，定理 4 より次の定理が導かれる．

定理 5 論理的問題 $\langle P,q\rangle$ の解集合 $L(P,q)$ は $sldGatom(P,q)$ に等しい．

定理 5により，論理的問題 $\langle P,q\rangle$ の解集合 $L(P,q)$ を SLD 導出で計算する方法が得られる．

[SLD 導出による解法]

1. r を q の述語とする．Δ の述語全体の集合 R に属さない述語 r'を選ぶ．ϕ を，r アトムが与えられて，その述語を r から r'に変える写像，G_1 を確定節 $(\phi(q) \leftarrow q)$ とする．
2. 計算規則 sel を選ぶ．
3. G_1 を sel に従って展開し，sel-SLD 成功列か sel-SLD 失敗列を得る．計算の途中で得られる未完列は必ず展開する．sel-SLD 未完列が残っており，計算が終結しないとき[7] は失敗である．

[7] 無限列が存在するときは，いくら展開しても計算は終結しない．

4. *sel*-SLD 成功列の最終ゴール節を すべて集めて単位節集合 F を得る.

5. 論理的問題 $\langle P, q\rangle$ の解集合 $L(P,q)$ を,
$L(P,q) = \bigcup_{(\phi(a)\leftarrow)\in F} rep(a)$
で求める. これは, F のなかの単位節 $(\phi(a)\leftarrow)$ から 得られる アトム a の基礎例全体の集合である.

6 木に基づく解法の比較

6.1 アンフォールド変換列の決定する木

節集合 $P' = P \cup \{G_1\}$ から始まり, アンフォールド変換を 有限回繰り返して得られるアンフォールド変換列に対して, 次のようにして 木を作る. アンフォールド変換列に出現する r' 節 をノードとする. そのアンフォールド変換列が得られる過程で, r' 節 C_a が 新しい r' 節 C'_a に展開されるとき, C_a から C'_a へのアークを作る. これらのノードとアークは 1つの有限の木を決定する[8].

その木のアークには,

1. 選択されたボディアトム

2. P から選ばれた確定節

の2項組をラベルとして付ける. このようにしてできるラベルつき木をアンフォールド変換列の木 と呼ぶ.

アンフォールド変換が1回追加されるたびに, 対応する木は成長する. すなわち, アンフォールド変換による解法とは, 木の根 G_1 から出発し, 各時点における木の葉のノードを展開して, 子ノードを得て木を拡張する操作を続けるものである. 木を拡張するためには, 各ノード (に対応するゴール節) のボディアトムを任意に選ぶ. 確定節が単位節でボディアトムが存在しない場合や, 選ばれたボディアトムに関するレゾルベントが存在しないノードは展開を停止することができる. すべての葉が展開を停止できる場合に, アンフォールド変換を終了して, 論理的問題の解を求めることができる.

6.2 木を基礎とした解法の比較

アンフォールド変換に基づく解法では, 各ノードに対応するゴール節が単位節でなければ, 任意のボディアトムを選ぶことができる. これに対して, あらかじめ定めた 計算規則でボディアトムを選択するという制限を加えれば, アンフォールド変換に基づく解法のサブクラスが得られる.

[8]ただし, 同じ節でも同じノードにしてはならない場合がある. そこで 厳密には, アンフォールド変換列で節が得られる履歴 (展開するときに P から選ばれた確定節の列) を 節とペアにしてノードとする.

容易にわかるように, アンフォールド変換に基づく解法を計算規則 *sel* で制限した解法は, *sel* によって導かれる SLD 導出による解法と "本質的に同じ" である. すなわち, レソルベントを求める計算の総和は 両者で相等しい.

7 枠組の拡大

7.1 解法のクラスの階層性

これまでの議論より,
$SLD \subset UNFOLD \subset ET$
という関係が成り立つ. すなわち, アンフォールド変換による解法 (UNFOLD) のクラスは, 等価変換による解法 (ET) における等価変換ルールの自由選択を放棄し, アンフォールド変換という特定の等価変換ルールだけを使うことに限定した解法のクラスである (4章). また, SLD 導出による解法 (SLD) のクラスは, アンフォールド変換による解法 (UNFOLD) におけるボディアトムの自由選択を放棄し, 計算規則という特殊な選択方式に限定した解法のクラスである (6章).

7.2 計算規則の問題の解消

すでに述べたように, SLD 導出が用いている計算規則には 問題がある. すなわち, SLD 導出に基づいて厳密に正当性を保証できる推論システムを構築しようとすれば, 計算規則を実現するために多くのコストがかかる. そのコストを低減するために, Prolog では ,「左から右」という単純な計算規則を用いている. これによって コストは 最小に抑えられるが, 計算の順序の柔軟性が損なわれることになり, 多くの制約充足問題が 事実上解けなくなる [4].

計算規則に発する この問題は, SLD 導出による解法を 等価変換による解法と比較することによって解決できる. 計算規則は 同じ "ゴール" に対して同じボディアトムを選択するという制約を表現するものであるが, アンフォールド変換による解法を見ればわかるように, それは不要な制限である. 本質的な制約は 「同じノードで同じボディアトムを選択する」という制約である. 計算規則の問題は, SLD 導出による解法が 計算規則という, 本質を反映していない機構を採用したために起こったことがわかる.

7.3 SLD 導出を越えて

ある問題を解くための解法のクラスが知られているとする. 実際にその問題を解くためには, その解法のクラスから, ある評価 (計算効率など) に基づいて, 最適な

解法を1つ選べばよい．この場合，より広い解法のクラスを知っているほど選択の幅が広いので，よりよい解法を選ぶことができる [2].

このことと，7.1 節の階層性：

$SLD \subset UNFOLD \subset ET$

の帰結として，SLD 導出による解法 (SLD) から，アンフォールド変換による解法 (UNFOLD) へ，そしてさらに等価変換による解法 (ET) へとクラスを拡大することによって，より多くの問題をより高速に解くために役立つことが予想できる．

実際，SLD 導出による解法 (SLD) から踏み出し，特にアンフォールド変換以外の等価変換ルールを使うことによって，計算を高速化を達成できる例は 数多い．たとえば [7] では，制約充足問題を解く場合，等価変換の枠組でアンフォールド変換以外の変換を援用することによって計算速度を著しく上げることができる例を示している．また，[3] では，アンフォールド変換だけを用いた自然言語理解システムに，アンフォールド変換以外の変換を追加することによって，システムを逐次的に高速化できることが示されている．

7.4 モジュラーな理論への移行

SLD においては，SLD レゾリューション全体について 1つの正当性の理論が証明されている．SLD の理論を 幾つかの部分の正当性の論理積に帰着することはできない．したがって，SLD ベースの理論では，SLD レゾリューションのやり方を改善するたびに，また一から正当性を証明しなければならない．これは，ドメインが難しくなればなるほど 大きな負担になり[9] 理論の発展を 困難にしている．

一方 等価変換による計算では，個々の等価変換の正当性を積み重ねることによって全体の計算の正当性が 保証される．これを 等価変換ルールの モジュラー性という．この結果，(アンフォールド変換に限らず) どんな等価変換ルールをも どんどん 追加投入して，解法を改善できる [3].

8 むすび

論理プログラミングとその拡張の研究においては，伝統的に SLD 導出，あるいは その 変種が 計算の基礎理論となってきた．しかし，SLD 導出の理論は 最適な理論とは いえない．それは，たとえば，実用システムを作る上で 非常に重要な，自由なボディアトム選択を 理論的に保証していない．本論文では，SLD 導出による解法のクラスを等価変換による解法のクラスに埋め込み，等価変換による解法を採用することによって，SLD 導出の理論の欠点を克服できることを示した．

従来の SLD 導出による解法の理論に比べて，等価変換に基づく解法の理論は，一般性が高く，より簡潔で明快である．等価変換を基礎とした 新たな理論体系を築くことが，計算機科学の更なる発展をもたらすものと期待される．

[9] これは 実際に [4] などで起こっている．

参考文献

[1] 赤間清，繁田良則，宮本衛市：論理プログラムの等価変換による問題解決の枠組，人工知能学会誌，Vol.12, No.2, pp.90–99 (1997).

[2] 赤間清，繁田良則，宮本衛市：論理的問題の等価変換による解法 (1)，その理論的基礎，人工知能学会誌，掲載予定

[3] 畑山満美子，赤間清，宮本衛市：等価変換ルールの追加による知識処理システムの改善，人工知能学会誌，Vol.12, No.6, pp.861–869 (1997)

[4] Van Hentenryck：Constraint Satisfaction in Logic Programming, The MIT Press (1989).

[5] Lloyd, J.W.: *Foundations of Logic Programming*, Second edition, Springer-Verlag (1987).

[6] Pettorossi, A. and Proietti, M. : Transformation of Logic Programs: Foundations and Techniques, *The Journal of Logic Programming*, Vol.19/20, pp.261–320 (1994).

[7] 吹田 慶子，赤間清，宮本衛市：等価変換による制約充足問題の解法，電子情報通信学会 ソフトウエアサイエンス研究会，SS 96-18, pp.1-8 (1996).