数理解析研究所講究録
第1964巻 2015年 133-151

正規言語のVariety Theoryと動的命題論理

浦本　武雄
京都大学・数理解析研究所

Takeo Uramoto*
Research Institute for Mathematical Sciences,
Kyoto University

概要

本稿は，(1) 動的命題論理と (2) 正規言語の Variety Theory という，理論計算機科学における異なる二つの主題に関わる．まず動的命題論理 (Propositional Dynamic Logic, PDL) とは，「非決定的プログラムの形式的検証」を目的とした様相論理で，1979 年に Fischer と Ladner により導入された．一方，正規言語の Variety Theory とは，端的に言うと「正規言語の組合せ的決定問題を，有限半群や有限半環 (semiring) の代数的な問題に還元する理論」であり，形式言語理論においてもその方法論は特に長い歴史を持つ．本稿は，動的命題論理に係る決定問題に，正規言語の Variety Theory の方法を応用することを主題とし，その手法を紹介する目的で書かれている．

1　はじめに

1.1　動的命題論理 (PDL) とは

動的命題論理 (Propositional Dynamic Logic, PDL) とは，1979 年に Fischer と Ladner [3] が導入した様相論理で，`while` プログラムを単純化した形式的表現 (regular program, cf. §2) の振舞いの形式的検証を目的としている．研究史的には，Hoare 論理 [6] や，様相論理を用いた Hoare 論理の Pratt による再定式化 [13] の文脈に属する．

PDL を含むプログラムの形式検証系では一般に，「プログラムが仕様通りに動作するか，そのコードを実行せず静的に検証すること」を目指す．システムの規模が大きくなると，プログラムが起こしうる動作パターンが組合せ的に爆発するため，それら全てを実際にシミュレーションすることは現実的でない．そのため，実際にプログラムを実行してテストするのではなく，出来る限り静的に動作検証を行いたい．一方で，Turing 完全なプログラム言語の詳細な検証 (停止性判定など) は，そもそも原理的に出来ないこと (決定不可能であること) が古くから数学的に分かっている．そのため，静的なプログラム解析を「如何にして**計算可能な範囲で**行うか」は非自明な課題となる．

結果，多くのプログラムの形式的検証での仕事は：

1. 対象とする問題を**理想化・単純化**する (仕様記述言語の syntax と semantics を定義する)；
2. 単純化した定式化で厳密に検証する (形式体系の完全性や決定可能性を証明する)；

の二つに分割される．純粋数学的な問題として取り扱いが可能であるのはあくまで 2. であるが，2. において良い結果 (決定可能性など) を得る為には，1. において適切な問題設定・形式体系定義を行う必要がある．

本稿で考察する動的命題論理 (以下，PDL) は，この意味で良い問題設定を与えている．実際 PDL は，

*uramoto@kurims.kyoto-u.ac.jp

Pratt が導入した動的論理 (Dynamic Logic, DL)[13] を大幅に単純化し，それにより良い決定可能性を獲得した：DL において推論対象とする論理式は一階述語論理式であり，その自然な意味論に対する妥当性は決定可能ではない．一方，PDL で対象とする論理式は量化子を含まず，それにより論理式の妥当性が (EXPTIME 完全な計算量で) 決定可能になる．記述言語の表現力で見ると，PDL は DL と比べて劣ると言わざるを得ないが，その代わりに「論理式の妥当性が決定可能」という重要な性質を獲得している点は無視できない．

Fischer と Ladner [3] による研究以降も，PDL の syntax の拡張および制限の研究が行われているなど，プログラムの形式検証の研究に PDL が与えた影響は大きい [5]．そのため本稿でも，PDL を基礎体系として選び，プログラム検証に係る決定問題を数学的に考察していく．

1.2 正規言語の Variety Theory を応用

PDL に関する研究は少なくないが，その中でも本稿の特色は，「PDL に関する決定問題群に対し**正規言語の Variety Theory** のアイデアを応用すること」にある．そしてそれにより「有限半群論で発達した代数的方法を，プログラム検証にまつわる組合せ的決定問題の解決に生かす」という一つのアプローチを生むことを目的としている．

正規言語の Variety Theory とは，概して言うと：

1. 正規言語の組合せ的性質；
2. 有限 monoid（単位的半群）の代数的性質；
3. 有限オートマトンの幾何的性質；

の間の対応関係を公理化した理論で，1975 年に Eilenberg [2] により創出された．応用上で特筆すべき点は，この対応関係を通して「正規言語の組合せ的性質を決定するアルゴリズム」を得られる場合があるという点である．この事実は，Variety Theory の黎明期に数多く観測され [14, 1, 15]，その後の正規言語の構造論の発展に大きな影響を与えた．現在も正規言語の Variety Theory を応用・拡張する試みが続けられており，当理論は今日の理論計算機科学においても，基礎 (foundation) を成す理論であると言ってよい．正規言語の Variety Theory そのものの概説は拙著 [18] に譲るとして，本稿はその応用に特化した話題を紹介する．

2 動的命題論理

2.1 背景

PDL の起源は，Hoare 論理 [6] の導入にさかのぼる．Hoare 論理は，メモリの状態を論理式により表現し，それがプログラム実行前後でどう変化するかを表現した「正しい」*Hoare triple* を形式的に推論することを目的とした論理体系である．Hoare triple とは一般に，

$$\{\phi\}p\{\psi\} \tag{1}$$

の形をした式を言う．ここで ϕ, ψ は論理式で，p はプログラムである．直感的に言うとこの Hoare triple は，「メモリ状態が ϕ を満たす時，プログラム p を実行して停止すれば，実行後のメモリ状態は ψ を満たす」ことを表わす．

Hoare triple が「正しい」ことの正確な定義は，その意味論に基づく．分かりやすい例としては例えば，プログラム p を：

```
begin
 i := 5;
 while i > 0 do
    x := x * i;
    i := i - 1
 od
end
```

としたとき，以下の Hoare triple は算術式の通常の意味論の元で正しい：

$$\{\mathtt{x} > 1\}\ p\ \{\mathtt{x} > 120\} \tag{2}$$

つまりこの Hoare triple は，「変数 x に 1 より大きい値が格納されている時，プログラム p を実行して停止すれば，その変数 x には必ず 120 より大きい値が格納されているはずである」ことを主張していて，

それが算術的に真であるため，この Hoare triple は正しい．一方，以下の Hoare triple は同じ意味論の元で偽となる：

$$\{x > 0\}\, p\, \{x > 120\} \qquad (3)$$

実際，例えば変数 x に値 1 が格納されている場合，$x > 0$ ではありながら，プログラム p の実行後は x にちょうど 120 の値が格納されているため，$x > 120$ とはならない．

勿論これらの真偽は手で確かめられる程度の単純なものである．しかしプログラムの規模や検証したい性質 (論理式 ϕ, ψ) が複雑になればなるほど，その正当性の検証をアドホックに実行することは現実的ではなくなる．そのため Hoare 論理は，(2) の様な「正しい Hoare triple」を**形式的に**推論できる公理系・推論規則を与えることを目標とする．その際もちろん，(3) の様な正しくない Hoare triple を推論してしまってはいけない．

Hoare 論理に関する研究は，今日，実用的なものから理論的なものまで様々ある．特に PDL との関連で基礎となるものは Pratt による研究 [13] で，彼は Hoare 論理を**様相論理** (*modal logic*) のアイデアに基づいて再定式化している．Pratt の研究も Hoare 論理と同じく，「形式論理によるプログラムの検証」を主題としている．しかし，その形式論理 (動的論理，DL) の syntax が様相論理式に基礎を置いている点に違いがある．

Hoare 論理が Hoare triple $\{\phi\}p\{\psi\}$ を推論対象とするのに対して，DL ではプログラム p を様相作用素 $\langle p\rangle, [p]$ に含む様相論理式を推論対象とする．例えば：

$$(x > 120)$$

が単に，「変数 x には 120 より大きい値が格納されている」ことを主張する論理式であるのに対して：

$$\langle p\rangle(x > 120)$$

という論理式は，「プログラム p を実行すると停止して，停止後の状態では変数 x の値が $(x > 120)$ となる」ことを表わす．

このように様相作用素 $\langle p\rangle$ は，それが作用する論理式に対して「プログラム p の実行後〜」という情報を付加する．そしてこの記法を使うと，Hoare triple (2) は：

$$(x > 1) \to \langle p\rangle(x > 120) \qquad (4)$$

と表せる．実際この論理式は，「変数 x に 1 より大きい値が格納されている $(x > 1)$ ならば $(\to)$，p を実行して停止すれば $(\langle p\rangle\sim)$，実行後の状態では変数 x には 120 より大きい値が格納されている $(x > 120)$」ことを表わしている．

PDL の syntax は，プログラムの動的な挙動も含めて論理式で記述するため，この DL の syntax に準拠している．しかし DL が一階述語論理式を使うのに対して，PDL の論理式は変数も量化子も含まない単純化されたものになっている．この単純化により，記述言語としての精密さを犠牲にしているが，一方で「論理式の妥当性が決定可能になる」という対価を得ている．

以下では PDL の定義 (§2.2) から始め，その具体的意味 (§2.3) および Fischer と Ladner 自身 [3] による主結果 (§2.4) を振り返る．

2.2 PDL の syntax と semantics

Syntax: 以下では有限集合 Π_0, Φ_0 を固定する．PDL の記述言語は，*regular program* と様相論理式からなり，それぞれの syntax は以下の相互再帰により定義される：

$$\phi \;::=\; 0 \mid \mathtt{b} \in \Phi_0 \mid \phi \cup \phi \mid \bar{\phi} \mid \langle p\rangle\phi \qquad (5)$$

$$p \;::=\; \phi? \mid \mathtt{p} \in \Pi_0 \mid p \cdot p \mid p \cup p \mid p^*. \qquad (6)$$

ここで 0 は `false` を表わす原始論理式で，$\phi?$ の形の regular program は，論理式 ϕ の**テスト** (*test*) と呼ばれる．以後，regular program 全体の集合を Π，論理式全体の集合を Φ と書く．定義から当然 $\Pi_0 \subseteq \Pi, \Phi_0 \subseteq \Phi$ となる．

注意 1. 以下では regular program における演算は，$* > \cdot > \cup$ の順で強い結合であるとする．例えば

$\Pi_0 = \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$ としたとき：

$$(\mathtt{p} \cdot (\mathtt{q}^*)) \cup \mathtt{r}$$

とは書かず，単に：

$$\mathtt{p} \cdot \mathtt{q}^* \cup \mathtt{r}$$

と書く．また，積 $\cdot$ の演算は省略する場合がある．例えば $\mathtt{pq}$ と書いたら，それは $\mathtt{p} \cdot \mathtt{q}$ を表わすとする．

注意 2. 簡単の為，論理式 $\phi, \psi \in \Phi$ および regular program $p \in \Pi$ に対して：

$$\phi \to \psi, \quad \phi \cap \psi, \quad [p]\phi,$$

と書いたら，それぞれ：

$$\bar{\phi} \cup \psi, \quad \overline{\bar{\phi} \cup \bar{\psi}}, \quad \overline{\langle p \rangle \bar{\phi}}$$

の略 (syntax sugar) であるとする．また以下では，$\mathtt{false}, \mathtt{true}$ と書いてそれぞれ，0 および $\bar{0}$ を表わすことがある．

Semantics: 各々の regular program や論理式そのものは単に形式的な記号列であるが，その「意味」は *Kripke frame* という，以下で定義される一種の有向グラフで表現される．

定義 1. Kripke frame とは以下の三つ組を言う：

1. **状態集合** Q;
2. **遷移関数** $\delta : \Pi_0 \to \mathcal{P}(Q \times Q)$;
3. **真偽値関数** $\tau : \Phi_0 \to \mathcal{P}(Q)$.

ここで $\mathcal{P}(X)$ は一般に，集合 X の部分集合の成す集合を表わす．

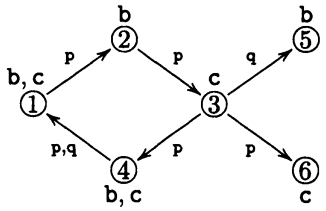
Kripke frame の意味は，三つ組 (Q, δ, τ) としての形式的な定義より，それを有向グラフで表現したものの方が視覚的に理解しやすい．一般に，Kripke frame (Q, δ, τ) が与えられたとき，状態集合 Q を頂点集合とする有向グラフを，次の様な規則で生成する：

1. $x, y \in Q, \mathtt{p} \in \Pi_0$ に対し $(x, y) \in \delta(\mathtt{p})$ のとき，頂点 $x, y \in Q$ の間に有向辺 $x \xrightarrow{\mathtt{p}} y$ を引く；
2. $x \in Q, \mathtt{b} \in \Phi_0$ に対し $x \in \tau(\mathtt{b})$ のとき，頂点 $x \in Q$ の上にラベル $\mathtt{b}$ をつける．

例 1. 例えば今，簡単のため $\Pi_0 = \{\mathtt{p}, \mathtt{q}\}$ および $\Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とする．もし Kripke frame (Q, δ, τ) が：

$$\begin{aligned} Q &:= \{1, 2, \cdots, 6\} \\ \mathcal{P}(Q \times Q) \ni \delta(\mathtt{p}) &:= \{(1,2), (2,3), (3,4), (3,6)\} \\ \mathcal{P}(Q \times Q) \ni \delta(\mathtt{q}) &:= \{(3,4), (3,5)\} \\ \mathcal{P}(Q) \ni \tau(\mathtt{b}) &:= \{1, 2, 4, 5\} \\ \mathcal{P}(Q) \ni \tau(\mathtt{c}) &:= \{1, 3, 4, 6\}. \end{aligned}$$

と定義されている時，この Kripke frame に対応して生成される有向グラフは以下のようになる：



(7)

実際この例では，例えば $(1,2) \in \delta(\mathtt{p})$ であるため，頂点 $1, 2$ の間に $\mathtt{p}$ でラベル付けされた辺 ① $\xrightarrow{\mathtt{p}}$ ② がある．また，$3 \in \tau(\mathtt{c})$ であるため頂点 ③ の上には $\mathtt{c}$ のラベルをつけている．

Kripke frame (Q, δ, τ) が与えられると，遷移関数 $\delta : \Pi_0 \to \mathcal{P}(Q \times Q)$ および真偽値関数 $\tau : \Phi_0 \to \mathcal{P}(Q)$ は，それぞれ regular program 全体の集合 Π および論理式全体の集合 Φ 上に自然に拡張される．

定義 2. (Q, δ, τ) を Kripke frame とする．この時，δ および τ の拡張 $\delta' : \Pi \to \mathcal{P}(Q \times Q), \tau' : \Phi \to \mathcal{P}(Q)$ が以下の相互再帰規則で定義できる：$\mathtt{p} \in \Pi_0$ および $\mathtt{b} \in \Phi_0$ 上では $\delta' = \delta$, $\tau' = \tau$ として，一般の regular

program $p \in \Pi$, 論理式 $\phi \in \Phi$ に対しては,

$$\begin{aligned}\delta'(\phi?) &:= \{(x,x) \in Q \times Q \mid x \in \tau'(\phi)\}.\\ \delta'(pq) &:= \delta'(p)\delta'(q)\\ \delta'(p \cup q) &:= \delta'(p) \cup \delta'(q)\\ \delta'(p^*) &:= \bigcup_{i=0}^{\infty} \delta'(p)^n\end{aligned}$$

ここで, 一般に二つの二項関係 $U, V \in \mathcal{P}(Q \times Q)$ に対して, その合成 $UV \in \mathcal{P}(Q \times Q)$ は:

$$\{(x,z) \in Q \times Q \mid \exists y \in Q.\ (x,y) \in U \wedge (y,z) \in V\}$$

で与えられる. また:

$$\begin{aligned}\tau'(0) &:= \emptyset\\ \tau'(\phi \cup \psi) &:= \tau'(\phi) \cup \tau'(\psi)\\ \tau'(\bar{\phi}) &:= Q \backslash \tau'(\phi)\end{aligned}$$

そして $\tau'(\langle p \rangle \phi) \in \mathcal{P}(Q)$ は以下で与えられる:

$$\{x \in Q \mid \exists x' \in Q.\ x \in \tau'(\phi) \wedge (x,x') \in \delta'(p)\}.$$

以下では Kripke frame (Q, δ, τ) が与えられた時, 状態 $x, y \in Q$ および regular program $p \in \Pi$ に対して, $(x,y) \in \delta'(p)$ となることを $x \xrightarrow{p} y$ と書く. また状態 $x \in Q$ および論理式 $\phi \in \Phi$ に対して, $x \in \tau'(\phi)$ となることを $x \models \phi$ と書き,「状態 x は論理式 ϕ を満たす」と言うことにする.

この $\delta'(p), \tau'(\phi)$ の形式的な定義だけでは, その幾何学的な意味が分かりづらいため, 例 1 の Kripke frame を使って, これらが具体的に何を計算しているのかを見てみる.

例 2. 例 1 と同様 $\Pi_0 = \{\mathtt{p}, \mathtt{q}\}, \Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とすると, 例えば $\mathtt{p} \cup \mathtt{q}$ は regular program である. $\delta'(\mathtt{p} \cup \mathtt{q})$ は $\delta(\mathtt{p}) \cup \delta(\mathtt{q})$ と定義されている為:

$$\begin{aligned}\delta'(\mathtt{p} \cup \mathtt{q}) = \{&(1,2),(2,3),(3,4),\\ &(4,1),(3,5),(3,6)\}\end{aligned}$$

となる. つまり $\delta'(\mathtt{p} \cup \mathtt{q})$ はグラフのラベル (今は $\mathtt{p}, \mathtt{q}$ しかない) に依らず, 有向辺 $x \to y$ でつながっている全ての状態ペア (x,y) を含む.

例 3. 一方, regular program $\mathtt{pq}$ に対し, $\delta'(\mathtt{pq}) = \delta(\mathtt{p})\delta(\mathtt{q})$ で定義されていた. 従ってグラフ (7) でそれを計算すると:

$$\delta'(\mathtt{pq}) = \{(3,1),(2,5)\}$$

となる. つまり, $x \xrightarrow{\mathtt{pq}} y\ (\Leftrightarrow (x,y) \in \delta'(\mathtt{pq}))$ となるのは, グラフの中で $x \xrightarrow{\mathtt{p}} \exists z \xrightarrow{\mathtt{q}} y$ というパスが存在する場合である.

例 4. また $\delta'(\mathtt{p}^*)$ は定義から:

$$\delta'(\mathtt{p}^*) = 1 \cup \delta(\mathtt{p}) \cup \delta(\mathtt{p})\delta(\mathtt{p}) \cup \cdots$$

となる為, $x \xrightarrow{\mathtt{p}^*} y$ となるのは $x = y$ の場合か, 或は $\mathtt{p}$ でラベル付けされた辺からなる有限長のパス $x \xrightarrow{\mathtt{p}} x_1 \xrightarrow{\mathtt{p}} x_2 \cdots \xrightarrow{\mathtt{p}} y$ が存在する場合である.

例 5. Kripke frame (7) において, 状態 $x \in Q$ が論理式 $\bar{\mathtt{b}}$ を満たす $(\Leftrightarrow x \in \tau'(\bar{\mathtt{b}}))$ のは定義から, $x \notin \delta(\mathtt{b})$ の時である. 従って:

$$\delta'(\bar{\mathtt{b}}) = \{3,6\}$$

となる. 言い換えると, $x \models \bar{\mathtt{b}}$ となるのは, 状態 x の上にラベル $\mathtt{b}$ が乗っていない時である.

例 6. 論理式 $\langle \mathtt{p} \rangle \mathtt{b}$ に対して $\tau'(\langle \mathtt{p} \rangle \mathtt{b})$ を考えると, 定義から (7) においては:

$$\tau'(\langle \mathtt{p} \rangle \mathtt{b}) = \{1,3,4\}$$

となる. つまり, $x \models \langle \mathtt{p} \rangle \mathtt{b}$ となる $(\Leftrightarrow x \in \tau'(\langle \mathtt{p} \rangle \mathtt{b}))$ のは, 状態 $y \in Q$ であって $x \xrightarrow{\mathtt{p}} y$ かつ $y \models \mathtt{b}$ となるものが存在する場合である.

例 7. 一般に論理式 $\phi \in \Phi$ と状態 $x \in Q$ に対して, regular program $\phi?$ は, $x \models \phi$ の時, またその時に限り状態遷移 $x \xrightarrow{\phi?} x$ を定める.

例 8. 一般に regular program $p, q \in \Pi$ と論理式 $\phi \in \Phi$ に対して, regular program $\phi?p \cup \bar{\phi}?q$ による遷移関係 $x \xrightarrow{\phi?p \cup \bar{\phi}?q} y\ (\Leftrightarrow (x,y) \in \delta'(\phi?p \cup \bar{\phi}?q))$ は:

$$x \xrightarrow{\phi?p \cup \bar{\phi}?q} y \quad \Leftrightarrow \quad \begin{cases} x \xrightarrow{p} y & \text{if } x \models \phi\\ x \xrightarrow{q} y & \text{if } x \models \bar{\phi}\end{cases}$$

となる.

2.3 Regular program の意味

これらの概念の実際的意味をもう少し把握するためには，具体的なプログラムを例に取って，PDL の syntax (regular program や論理式) や semantics (Kripke frame) が何を抽象化したものであるかを振り返ると良い．

まず PDL での regular program の役割を端的に言えば，「**プログラムが，どの命令を，どのパターンで呼び出しうるか**」を正規表現的に表現することにあると言える．実際，regular program の syntax は，文字列のパターンマッチで広く利用される**正規表現** [19] の syntax を (様相論理式との相互再帰で) 拡張したものになっている．そして「プログラムが呼び出しうる命令列のパターンを見ることで，プログラムの挙動を大まかに捉えよう」というのが PDL の基本姿勢である．

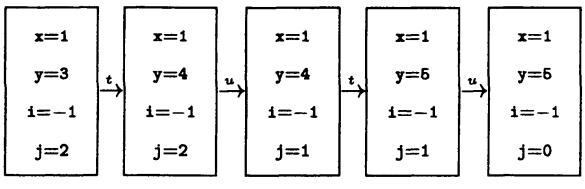
例えば今，while プログラム p を：

```
begin
 if i > 0 then x := x * i;
          else while j > 0 do
                 y := y + x;
                 j := j - 1;
               od
 fi
end
```

としてみる．この時，プログラム p を実行する際のメモリ状態に依存して，p を構成する基本命令である以下の代入命令 s, t, u が，一定のパターンで呼び出され実行されていく：

$$s \equiv [\mathtt{x} := \mathtt{x} * \mathtt{i}]$$
$$t \equiv [\mathtt{y} := \mathtt{y} + \mathtt{x}]$$
$$u \equiv [\mathtt{j} := \mathtt{j} - 1]$$

例えば，メモリ状態が x=1,y=3,i=-1,j=2 であるとき p を実行すると，次のような実行列になる：



一般にどのメモリ状態から始めても p は：

1. メモリ状態が論理式 (i > 0) を満たしていれば命令 $s \equiv [\mathtt{x} := \mathtt{x} * \mathtt{i}]$ を実行して終了し；
2. そうでなければ，論理式 (j > 0) を満たす限り命令列 $tu \equiv [\mathtt{y} := \mathtt{y} + \mathtt{x}; \mathtt{j} := \mathtt{j} - 1]$ を繰り返す．

ここで，命令の具体的な内容ではなく，命令が呼び出される動作パターンだけ見ると，if 文・while 文からなるプログラムの挙動は当然，「選択 (=if 文)」と「繰り返し (=while 文)」の使い方で決まる．PDL では regular program を使って，そういったプログラムの動作パターンを表現することが出来る．regular program は正規表現と同程度の簡明な syntax で定義されているが，プログラムの具体的内容ではなく，動作パターンのみに着目する目的では，その syntax で十分だといえる．

PDL の regular program を使って while プログラムの動作パターンを表現するには，(i) while プログラム内で使う基本命令の集合 Π_0 および基本論理式の集合 Φ_0 を固定して，(ii) while プログラムの構造に関して帰納的に while プログラムを regular program に書き替える，という作業を行えばよい．例えば：

```
begin
 if b then p;
      else q
 fi
end
```

という形の if 文のプログラムは，「メモリ状態が論理式 b を満たせば命令 p を実行し，さもなくば命令 q を実行する」というパターンで挙動する．そして，このパターンは以下の regular program で表現できる：

$$r = \mathtt{b?p} \cup \bar{\mathtt{b}}\mathtt{?q}$$

実際，例 8 でも見た様に，任意の Kripke frame (Q, δ, τ) および状態 $x, y \in Q$ に対して，$x \xrightarrow{r} y$ となるのは：

1. $x \models \mathtt{b}$ の場合は，$x \xrightarrow{\mathtt{p}} y$；
2. $x \models \bar{\mathtt{b}}$ の場合は，$x \xrightarrow{\mathtt{q}} y$；

となるときに他ならない．つまり regular program r は，“状態 x が論理式 b を満たす時は p を実行し，

さもなくば q を実行する" 上の if 文と同じ挙動を，全ての有向グラフ (Kripke frame) 上で実現する式になっている．

一方，以下のような while 文の場合：

```
begin
 while b do
  p;
 od
end
```

このプログラムは「論理式 b が成り立つ限り，命令 p を繰り返す」というパターンで動作する．このパターンは regular program では：

$$w \quad = \quad (\mathtt{b?p})^*\bar{\mathtt{b}}?$$

と表現できる．実際また定義 2 を振り返ると，この regular program w は任意の Kripke frame (Q,δ,τ) および任意の状態 $x,y \in Q$ に対して，$x \xrightarrow{w} y$ となるのは：

1. x が論理式 b を満たさず，$x = y$ の場合 (つまり何もしない場合) か；
2. x が論理式 b を満たし，論理式 b を満たす状態列 $x = x_0, x_1, \cdots, x_{n-1}$ および以下のパスがグラフ (Q,δ,τ) 内に存在し，かつ $y \models \bar{\mathtt{b}}$ となる場合

$$x = x_0 \xrightarrow{\mathtt{p}} x_1 \xrightarrow{\mathtt{p}} \cdots \xrightarrow{\mathtt{p}} x_n = y$$

に他ならないことが分かる．つまり regular program w は，"状態 x が論理式 b を満たさない場合は何もせず，さもなくば論理式 b を満たしている限り命令 p を繰り返す" という，上の while 文と同じパターンを，全ての有向グラフ (Q,δ,τ) 上で実現する式になっている．

より一般に，if 文・while 文からなるプログラムの動作パターンを表わす regular program は，そのプログラムの構造に関して帰納的に，上の if 文と while 文に対する書き換え規則を適用して行くことで得られる．例えば，上で例として挙げたプログラムは以下の様な形をしていた：

```
begin
 if b then s;
      else while c do
             t;
             u;
           od
 fi
end
```

すると，このプログラムの動作パターンを表現する regular program は，以下のようになる：

$$\mathtt{b?s} \cup \bar{\mathtt{b}}?(\mathtt{c?tu})^*\bar{\mathtt{c}}?$$

2.4 PDL の目標・過去の主結果

他の形式検証系と同様，PDL もプログラムの**静的な**動作検証を目指す．つまり各 regular program $p \in \Pi$ を各々の Kripke frame (Q,δ,τ) で解釈した時，その有向グラフの中でどんな遷移関係 $\delta'(p) \in \mathcal{P}(Q \times Q)$ になるかを，**実際に δ' を計算することなく**ある程度予測したい．

$\delta'(p)$ の構造は勿論 Kripke frame (Q,δ,τ) の構造に依存してはいるが，Kripke frame の選び方に依らず p **そのものの構造によって普遍的に決まる性質**も，少なからずある．Fischer と Ladner 自身は彼らの論文 [3] で，PDL の論理式で書ける regular program の普遍的性質の決定問題を議論し，その決定可能性を証明した．

例えば，while 文のプログラムに相当する regular program $w = (\mathtt{b?p})^*\bar{\mathtt{b}}?$ は，その動作パターンから分かるように，全ての Kripke frame (Q,δ,τ) および状態 $x \in Q$ において普遍的に，

> 状態 x が論理式 b を満たす時 (i.e. $x \models \mathtt{b}$)，
> regular program w を実行して状態 $y \in Q$ で停止すれば (i.e. $x \xrightarrow{w} y$)，その状態 y は
> 必ず論理式 $\bar{\mathtt{b}}$ を満たす (i.e. $y \models \bar{\mathtt{b}}$).

という性質を満たす．実際，$(\mathtt{b?p})^*\bar{\mathtt{b}}?$ に対応する while プログラム：

```
begin
 while b do
  p;
 od
end
```

は論理式 $\mathtt{b}$ を満たす限り停止しないため，それが停止したとすれば停止直後の状態 y は，$\mathtt{b}$ の否定である論理式 $\bar{\mathtt{b}}$ を満たしていなければならない．regular program $w = (\mathtt{b?p})^*\bar{\mathtt{b}}?$ のこの性質は，より形式的に言うと，PDL の以下の論理式 $\phi(w)$ が**妥当** (*valid*) であるということに他ならない：

$$\phi(w) \quad \equiv \quad \mathtt{b} \cap \langle w \rangle \mathtt{true} \rightarrow [w]\bar{\mathtt{b}} \qquad (8)$$

つまり，論理式 $\phi(w)$ は任意の Kripke frame (Q, δ, τ) および任意の状態 $x \in Q$ について：

$$x \models \phi(w) \qquad (9)$$

となる．

この例が示す様に PDL の論理式は，その中で様相 $\langle w \rangle, [w]$ として現れる regular program w の性質を表現している．そしてその論理式の妥当性は，その性質が Kripke frame に依らず普遍的に成り立つことを意味している．この関連で Fischer と Ladner が示したのは「PDL の任意の論理式 $\phi \in \Phi$ の妥当性が，EXPTIME 完全な計算量で決定可能である」という事実だった．このことから，もし regular program $p \in \Pi$ の性質が何らかの論理式 $\phi(p) \in \Phi$ で表現できる場合，その妥当性 (i.e. 普遍的に成り立つか否か) は，決定可能であることが従う．

論理式 ϕ の妥当性の決定は，ϕ の中に含まれる regular program の解釈 (i.e. $\delta'(p)$) を実際に計算すること無く (i.e. 静的に) 実行される．その意味で，Fischer と Ladner が示した結果は，regular program の静的な解析手法であると言える．この決定の為に Fischer と Ladner は論文 [3] で「充足可能な論理式 ϕ から有限 Kripke frame を構成する」手法を開発した．この手法は今日，Fischer-Ladner 閉包 (closure) などと呼ばれ，PDL 以外の様相論理の決定問題にも応用可能な汎用性のある方法であることが知られている (cf. [5])．

3 正規言語の Variety Theory へ

しかしながら，PDL の論理式の妥当性判定による，regular program の静的検証手法には限界もある．実際，PDL の論理式により表現できる regular program の性質は，必ずしも多くはない．その原因は端的に言うと，PDL の論理式の中では regular program が様相作用素 $\langle p \rangle, [p]$ として出現するのみであるため，表現できる性質が「p を実行した後〜」や「p を実行する前に〜なら」など，**regular program の実行前後の性質に限定されてしまう**ことに依る．そのため，regular program の考察したい性質が Kripke frame に依らず普遍的に成立するかどうかを，Fischer と Ladner のアルゴリズムによって決定できない場合がある．

本稿では，より踏み込んだ regular program の性質も決定できるようにするため，**正規言語の Variety Theory** に基づく手法を構築する．それにより，例えば「regular program の実行中に，決して命令 p を呼び出すことはない」など，**regular program の実行途中の性質**にも自然に言及できるようになる．結果的にこの決定手法を用いて，「与えられた regular program $p \in \Pi$ が "より良い形" をした等価な regular program に変換可能か」というタイプの決定問題が決定できるようになる (§4)．

しかし本稿の手法には代償がある．それは，考察する regular program のクラスを制限することである．一般の regular program は，テスト $\phi?$ として任意の様相論理式 $\phi \in \Phi$ を使って良い．しかし，本稿で対象とする regular program は，テストとして古典命題論理式のみ使うことを許す．つまり，テスト $\phi?$ の論理式 ϕ には，様相作用素 $\langle p \rangle, [p]$ を使ってはいけない (§3.2)．

元々この制限は，Kozen[8] が採用した制限で，この制限により regular program 同士の等価性が決定可能になる．もっと言うと等価性判定を，等式論理により完全に公理化することが出来，かつ正規言語の等価性判定に還元することが出来る [9]．本稿の手法は Kozen らによるこの還元 (を少し修正したもの) に基づいている [16, 17]．

3.1 Variety Theoryの基礎概念

Kozenによるregular programの制限の定義の前に，本稿で中心となる「正規言語のVariety Theory」の基礎概念を復習する．正規言語のVariety Theoryの歴史的な背景や現代的な研究状況については，[18]に概説がある．

正規言語のVariety Theoryの基本的なアイデアは，「正規言語Lに含まれる文字列パターンを，有限半群や有限半環(semiring)の代数的性質で特徴づけること」にある．この特徴付けが出来た場合，正規言語に係る決定問題を，有限代数の問題に帰着でき，それにより元の問題の決定可能性を示すことが出来る場合がある．本稿で使うのは，**正規言語のsyntactic semiring**という概念で，Polákにより基礎理論が作られた[12].

以下では有限アルファベットAに対して，A上の語(word)の有限集合$U=\{w_1,\cdots,w_n\}$全体の集合を，$\mathrm{F}(A^*)$と書く：

$$\mathrm{F}(A^*) \quad := \quad \{U \subseteq A^* \mid U \text{ は } A^* \text{ の有限部分集合 }\}.$$

また，$\mathrm{F}(A^*)$上に和と積を：

$$\begin{aligned} U+V &:= U \cup V \\ UV &:= \{uv \in A^* \mid u \in U \wedge v \in V\} \end{aligned}$$

で定義する．この時，これらの演算で$\mathrm{F}(A^*)$は半環となる．また，一つのword $w \in A^*$から成る元$\{w\} \in \mathrm{F}(A^*)$を単に$w \in \mathrm{F}(A^*)$と書く．その時，任意の$U=\{w_1,\cdots,w_n\} \in \mathrm{F}(A^*)$は，$U=w_1+\cdots+w_n$と書ける．空集合$\emptyset \in \mathrm{F}(A^*)$は半環$\mathrm{F}(A^*)$の和に関する零元である為，$0 \in \mathrm{F}(A^*)$と書く．また空語$\varepsilon$は積に関する単位元であるため，単に1と書くこともある．

定義 3 (syntactic semiring). Aを有限のアルファベットとし，$L \subseteq A^*$を言語とする．このとき，Lの**syntactic semiring**とは，半環$\mathrm{F}(A^*)$を以下のcongruence $\equiv_L$で割った商半環$S(L) := \mathrm{F}(A^*)/\equiv_L$を言う：$U,V \in \mathrm{F}(A^*)$に対して

$$U \equiv_L V \quad \Leftrightarrow \quad \begin{cases} \forall x,y \in A^*. \\ xUy \cap L \neq \emptyset \Leftrightarrow xVy \cap L \neq \emptyset \end{cases}$$

また自然な射影$\mathrm{F}(A^*) \twoheadrightarrow S(L)$を$\pi_L$と書く．

つまり$U=\{u_1,\cdots,u_n\}$と$V=\{v_1,\cdots,v_m\}$が$\equiv_L$に関して同値であるのは，任意のword $x,y \in A^*$に対して：

$$\begin{cases} 1 \leq \exists i \leq n. \\ xu_iy \in L \end{cases} \quad \Leftrightarrow \quad \begin{cases} 1 \leq \exists j \leq m. \\ xv_jy \in L. \end{cases}$$

となるときである．特にU,Vがそれぞれsingleton $u=\{u\}, v=\{v\} \in \mathrm{F}(A^*)$の時は，$u \equiv_L v$となるのは任意の$x,y \in A^*$に対して：

$$xuy \in L \quad \Leftrightarrow \quad xvy \in L$$

となる時に他ならない．以下では簡単の為，$U=w_1+\cdots+w_n \in \mathrm{F}(A^*)$の$\pi_L : \mathrm{F}(A^*) \twoheadrightarrow S(L)$による像$\pi_L(U) \in S(L)$を，単に$U \in S(L)$と書く．この時，$S(L)$が$\mathrm{F}(A^*)$の$\pi_L : \mathrm{F}(A^*) \twoheadrightarrow S(L)$による像である為，$S(L)$の任意の元は$w_1+\cdots w_n$ $(w_i \in A^*)$と書ける．つまり，$S(L)$は半環としてAの元で生成されている．

例 9. $A:=\{a,b\}$とし，言語$L \subseteq A^*$を：

$$L \quad := \quad \{a^n \in A^* \mid n \geq 0\} \tag{10}$$

とする．この時もしword $w \in A^*$がbを含めば，明らかにどんな$x,y \in A^*$でも$xwy \notin L$である．このことから：

$$w \equiv_L \emptyset \tag{11}$$

となることが分かる．つまり$w=\pi_L(w) \in S(L)$は，$S(L)$の零元$0=\pi_L(\emptyset) \in S(L)$となる．

例 10. 一般に言語$L \subseteq A^*$が禁止語(*forbidden word*)を持つ時，つまり，ある語$w \in A^*$が任意の$x,y \in A^*$で$xwy \notin L$となる時，$w \in S(L)$は半環$S(L)$の零元となる．逆に$w \in S(L)$が零元であるときは，wは言語Lの禁止語である．言い換えると，「wが$S(L)$で零元」という代数的性質は，「wがLの禁止語」という性質を特徴付けている．

例 11. $A=\{a,b\}$ とし言語 $L\subseteq A^*$ を：

$$L \;:=\; \{u_1\cdots u_n\in A^* \mid u_i = aa \text{ or } bb\,, n\geq 0\}$$

とする．このとき $S(L)$ において：

$$abb+bba \;=\; a$$

この $abb+bba$ と $a\in S(L)$ の等価性は，組合せ的に言うとより理解しやすい．$\equiv_L$ の定義から分かる通り $S(L)$ において $abb+bba=a$ が成り立つということは，$abb+bba\equiv_L a$ ということを意味する．つまり：

$$w=uav\in L \;\Leftrightarrow\; uabbv\in L \text{ または } ubbav\in L$$

このことから例えば，もし語 $w\in A^*$ が a を含み，かつ L の元であった場合は，w の中の a を abb または bba に置き換えた語 w' もまた，L の元でなければならない．

例 12. $A=\{a,b\}$ とし言語 $L\subseteq A^*$ を：

$$L \;:=\; \{(ab)^n \mid n\geq 0\}$$

とする．この時，その syntactic semiring $S(L)$ において：

$$(ab)^2 \;=\; ab$$

となる．つまり元 $ab\in S(L)$ は半環 $S(L)$ において idempotent である．実際，$S(L)=\mathrm{F}(A^*)/\equiv_L$ と $\equiv_L$ の定義を思い出すと，任意の $x,y\in A^*$ に対して：

$$xababy\in L \;\Leftrightarrow\; xaby\in L$$

となるからである．

このように $S(L)$ における等式 (e.g. $abb+bba=a$) は，L の中に出現する語のパターンを，より代数的に表現したものであり，場合によっては $S(L)$ の純代数的性質 (e.g. イデアルの自明性など) によって，L の組合せ的性質を簡明に特徴づけることが出来る場合がある．

このような対応関係は，特に**正規**言語の組合せ論で有効に生かされて来た．実際，言語 L の syntactic semiring $S(L)$ は，とくに L が正規言語の場合には，有限の半環となるからである：

命題 1 (Polák)．言語 $L\subseteq A^*$ の syntactic semiring $S(L)$ が有限半環である為の必要十分条件は，言語 L が正規言語であることである．

このことから，もし正規言語 L の組合せ的性質が $S(L)$ の代数的性質で特徴付けられた場合，$S(L)$ の有限性によって，元の正規言語 L の性質の決定可能性を証明できる場合がある．過去の正規言語の Variety Theory の研究でも，このようなアプローチで正規言語の組合せ的決定問題が解決されてきた．本稿での syntactic semiring の使い方も，このアプローチに準拠している．

3.2 制限された regular program

本節冒頭で言及したように，本稿で考察する regular program は，テスト $\phi?$ の使用を制限された regular program である．この制限は元々Kozen [8] が導入したもので，その制限により，(制限された) regular program の等価性判定を公理的に特徴付けることができ，かつ正規言語の等価性判定に還元することが出来るようになる [9].

本稿では，Kozen による還元を少し修正した還元 [16] を用いて，regular program に関する問題に正規言語の Variety Theory の手法 (syntactic semiring の構成) を応用する．それにより，例えば「与えられた regular program が，`while` プログラムから構成される regular program と等価になりうるか」の決定可能性などが示せるようになる．

定義 4 (制限された regular program)．有限集合 Π_0 および Φ_0 を固定する．このとき制限された regular program の syntax を，以下の帰納法で定義する：

$$\begin{aligned}\phi &::= 0 \mid \mathtt{b}\in\Phi_0 \mid \phi\cup\phi \mid \bar{\phi}\\ p &::= \phi? \mid \mathtt{p}\in\Pi_0 \mid pp \mid p\cup p \mid p^*.\end{aligned}$$

以下では制限された regular program のみを扱う為，単に regular program と言えば，制限されたものを指すことにする．

注意 3. 一般の regular program と異なるのは，論理式の syntax に様相論理式 $\langle p\rangle\phi$ を含まないことである．それにより，regular program のテスト $\phi?$ として現れるのは，古典命題論理式のみに制限されている．例えば，$\Pi_0 = \{\mathtt{p},\mathtt{q}\}, \Phi_0 = \{\mathtt{b},\mathtt{c}\}$ であるとき：

$$(\mathtt{b}?\mathtt{p})^*\bar{\mathtt{b}}? \qquad \mathtt{b}?\mathtt{c}?\mathtt{p} \cup \bar{\mathtt{c}}?\mathtt{q}$$

は制限された regular program であるが，一方で：

$$(\langle\mathtt{p}\rangle\mathtt{b})?\mathtt{q} \quad ([\mathtt{p}]\mathtt{b})?\mathtt{q} \cup (\langle\mathtt{p}\rangle\bar{\mathtt{b}})?\mathtt{p}$$

は制限された regular program ではない．

Kozen と Smith [9] は，このように制限された regular program の**等価性**を考察した．二つの regular program が等価であるというのは，直感的に言えば，それらが常に等しい振舞いをすることであるが，形式的には Kripke frame の言葉で次のように定義される．

定義 5 (等価性). p, q を regular program とする．このとき，p と q が**等価** (*equivalent*) であるとは，全ての Kripke frame (Q, δ, τ) に関して $\delta'(p) = \delta'(q)$ となることを言う．

例 13. regular program の等価性の意味を，具体的なプログラムによって理解するために，例えば以下の二つの while プログラムを考える：

```
# Program No.1
# Nested Loop
begin
 while b do
   p;
   while c do
    q;
   od
 od
end

# Program No.2
# No Nested Loop
begin
 if b then p;
           while b or c do
             if c then q;
                  else p;
             fi
           od
      else skip;
 fi
end
```

これらを前節で紹介した手続きで regular program に変換すると，それぞれ：

Program No.1 : $(\mathtt{b}?\mathtt{p}((\mathtt{c}?\mathtt{q})^*\bar{\mathtt{c}}?))^*\bar{\mathtt{b}}?$

Program No.2 : $\mathtt{b}?\mathtt{p} \cup ((\mathtt{b} \cup \mathtt{c})?(\mathtt{c}?\mathtt{q} \cup \bar{\mathtt{c}}?\mathtt{p}))^*\overline{\mathtt{b} \cup \mathtt{c}}?$

となるが，実はこれらの regular program は，上に定義した意味で等価になることが知られている [8].

この等価性の意味は，実際に「元のプログラムのインスタンスが，確かに同じ出力をすること」を具体例で確認することで，より良く理解することが出来る．その為に上の Program No.1 と Program No.2 の中で形式的な記号だった部分 (e.g. b, c や p, q の部分) に，具体的な論理式や関数をいれてみると，例えば以下のようになる：

```
# Program No.1 Instance
# Nested Loop
begin
 while i < 3 do
   x := x + i;
   i := i + 1;
   while j < 2 do
    y := y + 5;
    j := j + 1;
   od
 od
end

# Program No.2 Instance
# No Nested Loop
begin
 if i < 3 then x := x + i;
               i := i + 1;
               while i < 3 or j < 2 do
                 if j < 2 then y := y + 5;
                               j := j + 1;
                          else x := x + i;
                               i := i + 1;
                 fi
               od
      else skip;
 fi
end
```

実際に入れる関数・論理式は何でも良い．この時，例えばメモリ状態が `x=1,y=0,i=1,j=0` である際に，上の二つのプログラムを実際に実行してみると，最終的には共にメモリ状態が `x=4,y=10,i=3,j=2` で停止することが分かる．より踏み込んで言うと，「最初の状態 `x=1,y=0,i=1,j=0` から最後に停止する状態

x=4,y=10,i=3,j=2 までに，両プログラムがどの命令を実行したか」の実行列も一致していることが分かる．これが，両プログラムから構成される regular program が「等価である」ということの実際的な意味である．

一般に，プログラムを変換して得た regular program は，元のプログラムが「どのタイミングでどの命令を選択し，繰り返すか」のパターンを形式的に表現している．そのため変換後の regular program 同士の等価性は，元のプログラムの動作パターンの等しさを表わしている．

とくにテスト $\phi?$ が古典命題論理式 (定義 4 の ϕ) に制限されている場合，それらの間の等価性は (i) 等式論理により完全かつ健全に公理化可能，かつ (ii) 決定可能であることが Kozen と Smith [9] によって示されている．等価性を特徴付ける公理系は，正規表現の等価性を特徴付ける公理 [7] を拡張したものになっており，regular program 同士の等価性判定の計算複雑性も，正規表現の場合と同様 PSPACE 完全である．

以下では Kozen [8] による記法に合わせる為に，regular program における演算 $\cup$ を $+$；論理式 ϕ に対するテスト $\phi?$ を単に ϕ と書く．この約束の上で，(制限された) regular program の等価性を完全かつ健全に特徴づける公理系は，次で与えられる．

定義 6 (公理系 KAT). 論理式 ϕ, ψ に対して：

$$\phi\phi = \phi \qquad \phi\psi = \psi\phi$$
$$\phi + \bar{\phi} = 1 \qquad \phi\bar{\phi} = 0$$

また regular program s, t, u に対して：

$$s + (t + u) = (s + t) + u \qquad s \cdot (t \cdot u) = (s \cdot t) \cdot u$$
$$s + t = t + s \qquad s + s = s$$
$$s \cdot 1 = s \qquad 1 \cdot s = s$$
$$s \cdot (t + u) = s \cdot t + s \cdot u \qquad (s + t) \cdot u = s \cdot u + t \cdot u$$

また $s + t = t \Leftrightarrow s \leq t$ として：

$$s + t \cdot r \leq r \quad \Rightarrow \quad r \leq t^* \cdot s \tag{12}$$
$$s + r \cdot t \leq x \quad \Rightarrow \quad r \leq s \cdot t^* \tag{13}$$

この公理系を KAT と書き，二つの regular program p, q が KAT の公理を有限回適用して互いに移り合う時，KAT $\vdash p = q$ と書く．

Kozen と Smith [9] が示したのは，この公理系 KAT は，regular program の等価性に関して完全かつ健全であるということである：

定理 1 (Kozen-Smith). 任意の regular program p, q に対して，p, q が等価になる為の必要十分条件は，KAT $\vdash p = q$ となることである．

例 14. 例えば $\Pi_0 = \{\mathrm{p}, \mathrm{q}\}, \Phi_0 = \{\mathrm{b}, \mathrm{c}\}$ として，以下の二つの regular program を考える：

$$p \;=\; (\mathrm{bp\bar{b}})^*$$
$$q \;=\; 1 + \mathrm{bp\bar{b}}$$

これらは実は等価であることが分かる．実際，公理 (12) を適用すれば：

$$(\mathrm{bp\bar{b}})^* \;=\; 1 + \mathrm{bp\bar{b}} + (\mathrm{bp\bar{b}})(\mathrm{bp\bar{b}})(\mathrm{bp\bar{b}})^*$$

となることが分かる．しかし：

$$(\mathrm{bp\bar{b}})(\mathrm{bp\bar{b}}) \;=\; (\mathrm{bp})\bar{\mathrm{b}}\mathrm{b}(\mathrm{p\bar{b}})$$

かつ，KAT $\vdash \bar{\mathrm{b}}\mathrm{b} = 0$ であるため：

$$1 + \mathrm{bp\bar{b}} + (\mathrm{bp\bar{b}})(\mathrm{bp\bar{b}})(\mathrm{bp\bar{b}})^* \;=\; 1 + \mathrm{bp\bar{b}}$$

となる．従って，KAT $\vdash p = q$ が分かる．

3.3 正規言語の等価性判定への還元

上で見た様に，与えられた regular program が等価であるかは，公理系 KAT により完全かつ健全に特徴付けられる．しかしそれだけでは，regular program 間の等価性が決定可能であるかどうかは分からない．Kozen と Smith による仕事 [9] ではさらに，KAT $\vdash p = q$ となるか否かを，正規言語の等価性判定問題に帰着させることで，その決定可能性および計算複雑性を示している．

その中で「各 regular program p に対して効率的に正規言語 $G(p)$ を対応させる構成 $p \mapsto G(p)$」が基本的な役割を果たす．もし regular program がアルファベット Π_0, Φ_0 上で定義されている時，$G(p)$ はアルファベット $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の正規言語である．ただし $\overline{\Phi}_0$ は，各 $\mathtt{b} \in \Phi_0$ に対して決まる記号 $\bar{\mathtt{b}}$ からなる．

Kozen と Smith が与えた $G(p)$ の構成には，アルファベット $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の word のうち，*atom* と *guarded string* という概念が必要となる：

定義 7 (atom)．Φ_0 を $\Phi_0 = \{\mathtt{b}_1, \cdots, \mathtt{b}_N\}$ とする．このとき Φ_0 上の *atom* とは，$\Phi_0 \cup \overline{\Phi}_0$ 上の word $\alpha \in (\Phi_0 \cup \overline{\Phi}_0)^*$ であって以下の形をしたものを言う：

$$\begin{aligned} \alpha &= c_1 \cdots c_N; \\ &\quad \text{where } c_i \in \{\mathtt{b}_i, \bar{\mathtt{b}}_i\} \quad (1 \le i \le N). \end{aligned}$$

例えば，$\Phi_0 = \{\mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3\}$ の時，$\mathtt{b}_1\bar{\mathtt{b}}_2\bar{\mathtt{b}}_3$ や $\bar{\mathtt{b}}_1\mathtt{b}_2\bar{\mathtt{b}}_3$ は atom である．しかし $\mathtt{b}_1\mathtt{b}_2$ や $\mathtt{b}_2\mathtt{b}_1\mathtt{b}_3$ は atom ではない．

定義 8 (guarded string)．Π_0, Φ_0 上の *guarded string* とは，アルファベット $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の word であって，次の形をしたものを言う：

$$\alpha_0 \mathtt{p}_1 \alpha_1 \mathtt{p}_2 \cdots \alpha_{n-1} \mathtt{p}_n \alpha_n$$

ただしここで，各 i に対して $\mathtt{p}_i \in \Pi_0$ かつ α_i は Φ_0 上の atom である．以下では Π_0, Φ_0 上の guarded string 全体の集合を単に GS と書く．

Kozen と Smith は，各 regular program p に対してアルファベット $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の正規言語 $G(p)$ であって，特に guarded string のみからなるものを構成した (i.e. $G(p) \subseteq GS$)．この構成 $p \mapsto G(p)$ は「各 regular program p から，$G(p)$ を表わす正規表現が計算可能である」という意味で計算可能である．そして構成 $p \mapsto G(p)$ の以下の性質から，regular program p, q 同士の等価性 ($\Leftrightarrow$ KAT $\vdash p = q$) を，正規言語 $G(p), G(q)$ の等価性の判定問題に帰着させることが出来る．

定理 2 (Kozen-Smith)．任意の regular program p, q に対して：

$$\text{KAT} \vdash p = q \quad \Leftrightarrow \quad G(p) = G(q). \qquad (14)$$

が成り立つ．特に KAT $\vdash p = q$ は決定可能．

正規言語 $G(p)$ の構成は，regular program p の構造に関する帰納法で与えられる．以下，$\mathrm{At}(\Phi_0)$ と書いて Φ_0 上の atom 全体の集合を表わす．正規言語 $G(p)$ は，まず $\mathtt{p} \in \Pi_0$ および $\mathtt{b} \in \Phi_0$ に対しては：

$$\begin{aligned} G(\mathtt{p}) &:= \{\alpha \mathtt{p} \beta \mid \alpha, \beta \in \mathrm{At}(\Phi_0)\}; \\ G(\mathtt{b}) &:= \{\alpha \in \mathrm{At}(\Phi_0) \mid \mathtt{b} \in C(\alpha)\} \end{aligned}$$

そして一般には：

$$\begin{aligned} G(0) &:= \emptyset \\ G(\phi + \psi) &:= G(\phi) \cup G(\psi) \\ G(\bar{\phi}) &:= \mathrm{At}(\Phi_0) \backslash G(\phi) \\ G(p + q) &:= G(p) \cup G(q) \\ G(pq) &:= G(p) \diamond G(q) \\ G(p^*) &:= \bigcup_{n=0}^{\infty} \underbrace{G(p) \diamond \cdots \diamond G(p)}_{n} \end{aligned}$$

で与えられる．ただしここで，二つの guarded string $u = u'\alpha \in GS$ および $v = \beta v' \in GS$ に対して：

$$u \diamond v \quad := \quad \begin{cases} u'\alpha v' & \text{if } \alpha = \beta \\ \text{undefined} & \text{otherwise} \end{cases}$$

とし，$G(p) \diamond G(q) \subseteq GS$ を：

$$G(p) \diamond G(q) \quad := \quad \{u \diamond v \mid u \in G(p) \wedge v \in G(q)\}$$

とする．

例 15. 今，(順番込みで) $\Phi_0 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ とするとき：

$$\begin{aligned} G(\mathtt{b} \cap \mathtt{c}) &= \{\mathtt{abc}, \bar{\mathtt{a}}\mathtt{bc}\} \\ G(\bar{\mathtt{b}}) &= \{\mathtt{a}\bar{\mathtt{b}}\mathtt{c}, \mathtt{a}\bar{\mathtt{b}}\bar{\mathtt{c}}, \bar{\mathtt{a}}\bar{\mathtt{b}}\mathtt{c}, \bar{\mathtt{a}}\bar{\mathtt{b}}\bar{\mathtt{c}}\} \end{aligned}$$

となる．

例 16. 今，$\Pi_0 = \{\mathtt{p}, \mathtt{q}\}, \Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とするとき，`while` プログラム：

```
begin
 while b do
   p;
 od
end
```

に対応する regular program $r = (\mathtt{bp})^*\bar{\mathtt{b}}$ を考えると，正規言語 $G(r) \subseteq GS$ は：

$$\{\alpha_0 \mathtt{p} \alpha_1 \cdots \mathtt{p} \alpha_n \mid \mathtt{b} \in C(\alpha_i)\ (i \neq n), \bar{\mathtt{b}} \in C(\alpha_n)\}$$

となる．

注意 4. 大雑把に言うと，regular program p から構成される正規言語 $G(p)$ は，「p が実行しうる命令列の集合」を表現している [10]．例えば上の例では，`while` プログラムに対応する regular program の $G(r)$ を計算しているが，その結果式を見ると「論理式 b を満たす限り命令 p を実行し続ける」という `while` プログラムの動作パターンが，$G(r)$ の構造に自然に反映されていることが分かる．

このような経緯で，regular program p の等価性判定は，正規言語 $G(p)$ の等価性判定に帰着される．この正規言語 $G(p)$ は regular program p が「どのような命令を呼び出すか」を表わし，p の挙動について十分な情報を含んでいる．

本稿ではさらに「正規言語の組合せ的性質を，その syntactic semiring の代数的性質に帰着させる」正規言語の Variety Theory の方法を使って，regular program の挙動を代数的に調べたい．この目的の為に，正規言語 $G(p)$ の構成を修正して新しい正規言語 $W(p)$ を作り，その syntactic semiring $S(W(p))$ を考える．この syntactic semiring は等式論理 KAT の有限モデルとなり，本稿の手法上，中心的な役割を果たす．

4 有限モデルの構成

前節でしばらく，元の問題 (つまり regular program の静的コード検証) から遠ざかっていたため，ここでもう一度，我々が何を目指しているかを思い出しておく．

端的に言えば regular program p が与えられた時，それが「どのような振舞いをするか」を，p の組合せ的な構造だけから決定したい．より正確に言うと，p が各 Kripke frame (Q, δ, τ) において，どのような遷移関係 $\delta'(p) \in \mathcal{P}(Q \times Q)$ を定めるかを，実際に $\delta'(p)$ を計算することなく p の情報だけから，ある程度予測したい，

本稿では正規言語 $G(p)$ を少し修正した正規言語 $W(p)$ の syntactic semiring の構造を見ることで，この問題に答える方法を与える．$G(p)$ を修正する理由は，簡単に言えば (i) $G(p)$ の syntactic semiring が，KAT の公理を満たさず，KAT のモデルとならないことにある (cf. [16])．そしてこの事実は (ii) 正規言語 $G(p)$ に関する決定問題を，その syntactic semiring の代数的問題に帰着させる際に障害となる．その為 [16] では，$G(p)$ の構成を修正して新しい正規言語 $W(p)$ を作り，(i) $W(p)$ の syntactic semiring が KAT の有限モデルとなりつつ，(ii) $G(p)$ の完全性 (定理 2) と同様の完全性が成り立つ様にしている．つまり：

定理 3 ([16])．任意の regular program p, q に対して：

$$\mathrm{KAT} \vdash p = q \quad \Leftrightarrow \quad W(p) = W(q) \qquad (15)$$

が成り立つ．とくに，$W(p)$ の syntactic semiring は，KAT の有限モデルとなる．

以下この節では，正規言語 $W(p)$ の構成 (§4.1)；有限モデル $K(p)$ の構成と利用法 (§4.2)；および決定問題への応用 (§4.3) を紹介する．

4.1 正規言語 $W(p)$

正規言語 $G(p)$ は，各 regular program p に対して構成され，guarded string と呼ばれる特殊な word から成っていた．一方これから構成する正規言語 $W(p)$ は，*weakly guarded string* と呼ばれる word から成る．その構成の為に，まず atom を弱めた概念である *consistent test* の概念を定義する．以下では word $u =$

$a_1 \cdots a_n \in A^*$ に対して，$C(u) := \{a_1, \cdots, a_n\} \subseteq A$ と書く (e.g. $C(\mathtt{abba}) = \{\mathtt{a}, \mathtt{b}\}$).

定義 9 (consistent test). $\Phi_0 = \{\mathtt{b}_1, \cdots, \mathtt{b}_N\}$ とする．この時，Φ_0 上の *consistent test* とは，アルファベット $\Phi_0 \cup \overline{\Phi}_0$ 上の word $\alpha' \in (\Phi_0 \cup \overline{\Phi}_0)^*$ であって，各 i に対して，$\mathtt{b}_i \in C(\alpha')$ かつ $\bar{\mathtt{b}}_i \in C(\alpha')$ とはならないものを言う．

例 17. 例えば $\Phi_0 = \{\mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3\}$ の時，$\mathtt{b}_2\mathtt{b}_1\mathtt{b}_1\bar{\mathtt{b}}_3\mathtt{b}_2$ や $\mathtt{b}_1\bar{\mathtt{b}}_3\mathtt{b}_1$ は consistent test である．一方，$\mathtt{b}_1\mathtt{b}_3\mathtt{b}_2\bar{\mathtt{b}}_3$ は consistent test ではない．

一般に，atom は consistent test の一種である．しかし一般の consistent test は atom とは異なり，各 $\mathtt{b}_i \in \Phi_0$ が任意の順序で何回現れても (現れなくても) 良い．

$W(p)$ の構成に使う weakly guarded string は，guarded string 中の atom を consistent test に置き換えたものであり，guarded string を一般化している．

定義 10 (weakly guarded string). Π_0, Φ_0 上の *weakly guarded string* とは，$\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の word であって，以下の形をしたものを言う：

$$\alpha'_0\mathtt{p}_1\alpha'_1\mathtt{p}_2 \cdots \alpha'_{n-1}\mathtt{p}_n\alpha'_n$$

ただしここで，各 i に対して $\mathtt{p}_i \in \Pi_0$ かつ α'_i は Φ_0 上の consistent test である．以下では Π_0, Φ_0 上の weakly guarded string 全体の集合を単に WGS と書く．

例 18. $\Pi_0 = \{\mathtt{p}, \mathtt{q}\}, \Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とする．このとき，例えば以下は Π_0, Φ_0 上の weakly guarded string である：

$$\mathtt{bpqcb}, \quad \mathtt{bb}, \quad \mathtt{pbcbpc}$$

また，空語 ε は Φ_0 上の consistent test であるので，Π_0 上の任意の word $\mathtt{p}_1 \cdots \mathtt{p}_n$ は Π_0, Φ_0 上の weakly guarded string である．

今，Π_0, Φ_0 上の regular program p が与えられた時，Π_0, Φ_0 上の weakly guarded string から成る正規言語 $W(p) \subseteq WGS$ を，$G(p)$ から自然に作ることができる．

そのためにまず，guarded string $u \in GS$ と weakly guarded string $v \in WGS$ の間の関係 $v \sqsubseteq u$ を，以下で定義する：$u = \alpha_0\mathtt{p}_1\alpha_1\mathtt{p}_2 \cdots \alpha_{n-1}\mathtt{p}_n\alpha_n \in GS$ および $v = \alpha'_0\mathtt{q}_1\alpha'_1\mathtt{q}_2 \cdots \alpha'_{m-1}\mathtt{q}_m\alpha'_m \in WGS$ の時，

$$v \sqsubseteq u \quad \Leftrightarrow \quad \begin{cases} n = m \\ \mathtt{p}_i = \mathtt{q}_i \\ C(\alpha'_i) \subseteq C(\alpha_i) \end{cases}$$

この記号を使って，$W(p)$ を以下で定義する：

$$W(p) \quad := \quad \{v \in WGS \mid \exists u \in G(p).\, v \sqsubseteq u\}$$

この時 $G(p)$ と同様，$W(p)$ はアルファベット $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の正規言語になる．そして構成 $p \mapsto G(p)$ が計算可能であったのと同様に，$p \mapsto G(p) \mapsto W(p)$ も計算可能になる．つまり regular program p から，$W(p)$ を表現する $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の正規表現は計算可能である [16].

例 19. $\Pi_0 = \{\mathtt{p}, \mathtt{q}\}, \Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とする．この時：

```
begin
 while b do
   p;
 od
end
```

に対応する regular program $r = (\mathtt{bp})^*\bar{\mathtt{b}}$ を考える．この regular program r から構成される $W(r)$ を具体的に計算すると：

$$\{\alpha'_0\mathtt{p}\alpha'_1 \cdots \mathtt{p}\alpha'_n \mid \bar{\mathtt{b}} \notin C(\alpha'_i)\ (i \neq n),\ \mathtt{b} \notin C(\alpha_n)\}$$

となる．

$G(p)$ と比べて特に重要な $W(p)$ の性質は，「任意の regular program p に対して，$W(p)$ の syntactic semiring $K(p) := S(W(p))$ が KAT の有限モデルとなること」である．そして $K(p)$ が KAT の有限モデルになる事実を使って，p に関する性質を，$K(p)$ の代数的性質で特徴付け，結果的に p の性質の決定可能性を示すことが出来る場合がある．

4.2 有限モデル $K(p)$

KAT のモデルとは一般に，組 $\langle K, B\rangle$ であって (i) K が Kleene 代数 [7]，(ii) $B \subseteq K$ が部分ブール代数となるものを言う．各 regular program p に対して「$K(p) = S(W(p))$ が KAT のモデルとなる」というのは，$K(p)$ には自然な部分ブール代数が存在していることを意味する．

$K(p)$ は定義から，アルファベット $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ 上の正規言語 $W(p)$ の syntactic semiring $S(W(p))$ である．従って，$K(p)$ は半環として $\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0$ で生成されている (i.e. 半環の全射準同型 $\pi_{W(p)} : \mathrm{F}((\Pi_0 \cup \Phi_0 \cup \overline{\Phi}_0)^*) \twoheadrightarrow K(p)$ が存在する)．このうち特に，$\Phi_0 \cup \overline{\Phi}_0$ で生成されている部分代数を $B(p)$ と置くと，実は $B(p)$ はブール代数となっていることが分かっている (cf. [17])．この二つの代数 $\langle K(p), B(p)\rangle$ は，等価な regular program に対しては同型になるため，その意味で regular program の一種の代数的不変量を与えている．

この代数の組 $\langle K(p), B(p)\rangle$ の構造は，元の regular program p の情報を反映している．今特に興味があるのは，$\langle K(p), B(p)\rangle$ の構造を見ることで，「p が各 Kripke frame (Q, δ, τ) で，どのような遷移関係 $\delta'(p) \in \mathcal{P}(Q \times Q)$ を定めるか」についての情報を得ることである．この段階に来て重要になるのは，$\langle K(p), B(p)\rangle$ が KAT のモデルになるという性質である．実際，その性質のおかげで，各様相論理式 $\phi \in \Phi$ から効率的に $B(p)$ の元 $\rho_p(\phi) \in B(p)$ を定義することが出来，その $\rho_p(\phi)$ の値の情報で p の性質を知ることが出来る[1]：

定義 11. p を Π_0, Φ_0 上の (制限された) regular program とする．この時，(任意の) regular program $r \in \Pi$ と論理式 $\phi \in \Phi$ に対して，以下の規則で帰納的に $\rho_p(r) \in K(p)$ および $\rho_p(\phi) \in B(p)$ を定めることが出来る：

$$
\begin{aligned}
\rho_p(0) &:= 0 \\
\rho_p(\mathtt{b}) &:= \mathtt{b} && (\mathtt{b} \in \Phi_0) \\
\rho_p(\phi + \psi) &:= \rho_p(\phi) + \rho_p(\psi) \\
\rho_p(\bar{\phi}) &:= \overline{\rho_p(\phi)} \\
\rho_p(\langle q\rangle\phi) &:= \bigcap\{a \in B(p) \mid \bar{a}\rho_p(q)\rho_p(\phi) = 0\} \\
\rho_p(\mathtt{q}) &:= \mathtt{q} && (\mathtt{q} \in \Pi_0) \\
\rho_p(q + r) &:= \rho_p(q) + \rho_p(r) \\
\rho_p(q^*) &:= \rho_p(q)^*
\end{aligned}
$$

ここで $\rho_p(q)^* \in K(p)$ は，$\rho_p(q) \in K(p)$ の Kleene 閉包である．

特にもし，様相論理式 $\phi \in \Phi$ に対して，$\rho_p(\phi) \in B(p)$ がブール代数 $B(p)$ の最大元 $1 = \mathtt{true}$ と一致する時，以下では：

$$K(p) \models \phi \tag{16}$$

と書き，ϕ は $K(p)$ で**妥当** (*valid*) であるということにする．

勿論，regular program p の構造によって，どの論理式 $\phi \in \Phi$ が $K(p)$ で妥当であるかは異なる．逆に言えば「どの論理式 $\phi \in \Phi$ が $K(p)$ で妥当であるか」によって，regular program p の性質を捉えることが出来る．

例 20. $\Pi_0 = \{\mathtt{p}, \mathtt{q}\}, \Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とし，例 19 の while プログラム：

```
begin
 while b do
   p;
 od
end
```

に対応する regular program $r = (\mathtt{bp})^*\bar{\mathtt{b}}$ について，$\langle K(r), B(r)\rangle$ を考えると：

$$K(r) \models \langle \mathtt{p}\rangle\mathtt{true} \leftrightarrow \mathtt{b}$$

となる．

これは直感的に言うと，「while プログラム r の実行中に，もし命令 p が呼び出されるとすると，その直

[1] この定義の際に，$\Phi_0 \cup \overline{\Phi}_0$ で生成される $K(p)$ の部分代数 $B(p)$ がブール代数になること (特に $B(p)$ の元が補元を持つこと) を使う．$G(p)$ の syntactic semiring では，$W(p)$ の場合と異なり，$\Phi_0 \cup \overline{\Phi}_0$ で生成される部分代数がブール代数にならない為，同様の定義が出来ない．

なお，ここでの $\rho_p(q) \in K(p)$ や $\rho_p(\phi) \in B(p)$ の構成は，$K(p)$ のイデアルでパラメータ付けられた・より一般的な構成の特殊な例である．簡単のため，本稿では一般的な場合は扱わない．

前の状態は，論理式 b が満たされている状態である」ことを示している．そしてこれは単に直感的な話ではない．実際もし一般に regular program r に対し，論理式 $\langle \mathtt{p} \rangle \mathtt{true} \leftrightarrow \mathtt{b}$ が $K(r)$ において妥当である時，各 Kripke frame (Q, δ, τ) で r が定めるグラフ上のパスが p を含めば，その直前の状態は論理式 b を満たすことが分かる．

例 21. $\Pi_0 = \{\mathtt{s}, \mathtt{t}, \mathtt{u}\}, \Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とし，以下の if 文：

```
begin
 if b then s;
      else t;
 fi
end
```

に対応する regular program $r = \mathtt{bs} + \bar{\mathtt{b}}\mathtt{t}$ を考えると：

$$K(r) \models \langle \mathtt{u} \rangle \mathtt{false}$$

となる．これは「r の実行中に命令 u が呼び出されることはない」事実を捉えている．実際，任意の Kripke frame (Q, δ, τ) で r が定めるグラフ上のパスは，u でラベル付けられた有向辺 $x \xrightarrow{\mathtt{u}} y$ を含むことはない (現れるラベルは s か t のみ)．

このように regular program p から構成される $\langle K(p), B(p) \rangle$ の構造は，「それがどんな論理式を満たすか $K(p) \models \phi$」により，元の p の挙動についての情報を与えてくれる．そして各 ϕ の $B(p)$ における値 $\rho_p(\phi) \in B(p)$ は，ϕ の構造に関して帰納的に定義されているため計算可能である．そのため，$K(p) \models \phi$ か否かは，全ての p 及び ϕ について決定可能である．

4.3 決定問題への応用

与えられた regular program p および論理式 $\phi \in \Phi$ について，ϕ が $K(p)$ で妥当であるか否かは決定可能である．従ってもし p に関して興味のある性質 X が何らかの論理式 ϕ によって特徴づけられた場合 (i.e. p が X を満たす $\Leftrightarrow K(p) \models \phi$)，その性質 X が決定可能であることが従う．本稿の最後として，ここでは，そのような応用が実際に可能であることについて紹介する．

ここで考察する問題は特に，「与えられた regular program p が，何らかの while プログラムから構成される regular program と等価であるか」の決定可能性である．そして「何らかの while プログラムから構成される regular program と等価である」という性質 X が，上述の意味で，ある論理式 ϕ によって特徴づけられることを見ることで，X の決定可能性を示す．

これまでも何度か例で出て来ているように，if 文と while 文からなるプログラムから regular program を一定の手順で構成できた．例えば例 19 では：

```
begin
 while b do
   p;
 od
end
```

というプログラムを考えたが，これは regular program $(\mathtt{bp})^*\bar{\mathtt{b}}$ に変換される．しかし一般に全ての regular program がこのような形で現れる訳ではない．実際，例えば $\Pi_0 = \{\mathtt{p}, \mathtt{q}\}, \Phi_0 = \{\mathtt{b}, \mathtt{c}\}$ とした時：

$$\mathtt{p} + \mathtt{q} \qquad \mathtt{p}^*$$

などは，どんな while プログラムからも構成できないばかりか，そのような regular program と等価にすらならない[2]．

そのため「与えられた regular program が，何らかの while プログラムから構成される regular program と等価に成りうるか」の決定可能性は非自明な問題となる．さらに言うと，ただ決定問題として非自明なだけではない．実際，静的検証の複雑性に関して，while プログラムから構成される regular program (以下 *well-structured* regular program) には長所があることが知られている：well-structured regular program についての Halpern と Reif [4] の研究によると，一般の regular program を含む様相論理式の妥当性の決定には EXPTIME 完全な計算量が必要である一方，well-structured な regular program

[2] while プログラムから構成される regular program は常に決定的 (deterministic) な遷移を定めるのに対して，$\mathtt{p} + \mathtt{q}$ や $\mathtt{p}^*$ は非決定的 (nondeterministic) な遷移を定めることから直ちに分かる．

のみを含む様相論理式の妥当性の決定はPSPACE完全な計算量でよい.

このことは「もし一見してwell-structuredでないregular program pが，実はwell-structuredなものと等価であるならば，それと取り替える方が検証の為にも有利である」ことを意味している．本稿の手法は実は，そのようなwell-structured regular programの存在が決定可能であることを示すのに応用できる．実際この小節の冒頭でも言及したように,「与えられたregular program pに対して，等価なwell-structured regular programが存在するか否か」という性質Xを，論理式で特徴付けることができる．その論理式は見た目は分かりづらいが，簡単に表現すると「pが決定的な挙動をする」ことを特徴づける論理式になっており，その論理式を$K(p)$が満たす時，pには実はwell-structuredなregular programで等価なものが存在することが分かる：

定理 4. pをΠ_0, Φ_0上のregular programとする．この時，pと等価なwell-structured regular programが存在するための必要十分条件は，以下が成り立つことである：

$$K(p) \models \bigcap_{\mathtt{r}\in\Pi_0} [p\mathtt{r}]\mathtt{false} \cap \bigcap_{\mathtt{r}\neq\mathtt{s}\in\Pi_0} ([\tau(p)\mathtt{r}]\mathtt{false} \cup [\tau(p)\mathtt{s}]\mathtt{false})$$

ここでτは，(ある計算可能な仕方で)regular programを変換する変換である．

この特徴付けから結果的に，等価なwell-structured regular programが存在するか否かは，決定可能であることが従う．

系 1. 与えられたregular program pに対して，それと等価なwell-structured regular programが存在するか否かは，決定可能である．

注意 5. より踏み込んで言うと，もしpに対して，等価なwell-structured regular program qが存在するとき，それをpから自動生成するアルゴリズムを与えることも出来る．しかし本稿ではそれは扱わない．

5 おわりに

本稿の主題の一つである正規言語のVariety Theoryは，1965年のSchützenbergerの研究[14]に起源を持ち，形式言語理論においても最も長い歴史のある深淵な理論の一つと言って良い．近年では，[18]でも概説したように，可換代数やガロア圏の理論との接点も明らかになりつつあり，正規言語のVariety Theoryの理論的な基礎については，既に十分に理解されていると言える．その現代的な視点から見ると，正規言語のVareity Theoryの根幹にある原理は,「正規言語と有限monoid (或は有限semiring)，および有限オートマトンの間の双対性」であると簡潔に表現することが出来る．

この双対性は単なる理論的お飾りではない．実際，その双対性を通して正規言語を眺めることで，正規言語に関する多くの有益な情報が得られるからだ．本稿では特にこの点 (応用可能性) を強調するため，正規言語のVariety Theoryの方法をより計算論的・数理論理学的な決定問題へ応用することに専念して，その結果を紹介した．しかし直感的な解説に紙数を費やしているため，個々の命題に証明を与えることが出来なかった．そのため詳しい証明やより正確な定義については，所々で引用した原論文をあたる必要がある．

PDLや関連する話題については，Kozenらによる教科書[5]が詳しい．等式論理KATに関しては，[8, 9]が原典であり，正規言語のVariety Theoryの手法の応用については，[16, 17]で詳しい証明をみることが出来る．しかし定理4などの証明は[16, 17]に含まれていない為，近く別の所で与える．Variety Theory自体の文献や近年の研究については，[18]にも概説してあるが，上で挙げた例だけで言えば[1, 14, 15]が，Variety Theoryの最初期を象徴する古典的結果である．またオンラインで入手できる文献としては，[11]が正規言語理論の基礎的な話題を，教科書的に網羅している．

本稿はこれら異なる領域の研究に，一つの文脈を作ることを意図して書かれている．本稿が，理論と実践の間の研究交流へ寄与することを願っている．

参考文献

[1] Janusz Brzozowski and Imre Simon. Characterizations of Locally Testable Languages. In *12th Annual Symposium on Switching and Automata Theory, East Lansing, Michigan, USA*, pages 166–176, 1971.

[2] Samuel Eilenberg. *Automata, Languages and Machines.* Academic Press, Inc., Orland, FL, USA, 1976.

[3] Michael Fischer and Richard Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[4] Joseph Halpern and John Reif. The Propositional Dynamic Logic of Deterministic, Well-structured Programs. *Theoretical Computer Science*, 27:127–165, 1983.

[5] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic.* MIT Press, Cambridge, MA, USA, 2000.

[6] C.A.R. Hoare. An Axiomatic Basis for Computer Programming. *Communications of the ACM*, 12(10):576–580, 1969.

[7] Dexter Kozen. A Completeness Theorem of Kleene Algebras and the Algebra of Regular Events. *Information and Control*, 110:366–390, 1994.

[8] Dexter Kozen. Kleene Algebra with Tests. *ACM Trans. Program. Lang. Syst.*, 19:427–443, 1997.

[9] Dexter Kozen and Frederick Smith. Kleene Algebra with Tests: Completeness and Decidability. In *Proc. 10th Int. Workshop Computer Science Logic (CSL '96)*, pages 244–259. Springer-Verlag, 1996.

[10] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In *Handbook of Theoretical Computer Science*, pages 789–840. Elsevier, 1990.

[11] Jean-Éric Pin. Mathematical Foundations of Automata Theory. `http://www.liafa.jussieu.fr/~jep/PDF/MPRI/MPRI.pdf`, 2015.

[12] Libor Polák. Syntactic semiring of a language. In *Mathematical Foundations of Computer Science 2001*, pages 611–620. Springer, 2001.

[13] Vaughan Pratt. Semantical Considerations on Floyd-Hoare Logic, 1976.

[14] M.P. Schützenberger. On finite monoids having only trivial subgroups. 8:190–194, 1965.

[15] Imre Simon. Piecewise Testable Events. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, pages 214–222, 1975.

[16] Takeo Uramoto. A Modified Completeness Theorem of KAT and Decidability of Term Reducibility. In *Proc. 14th International Conference on Relational and Algebraic Methods in Computer Science*, pages 83–100, April 2014.

[17] Takeo Uramoto. Canonical Finite Models of Kleene Algebra with Tests. *Journal of Logical and Algebraic Methods in Programming*, in press.

[18] 浦本武雄. 双対性による正規言語の Variety Theory. **数理解析研究所講究録**, 1915:100–112, 2014.

[19] 新屋良磨 鈴木勇介, 高田謙. **正規表現技術入門 最新エンジン実装と理論的背景**. 技術評論社, 2015.