計算資源使用法検証における計算資源の仕様と実際の使用法との間の適合性検証アルゴリズム

岩間 太[†] 五十嵐 淳[††] 小林直樹[†]

ファイルやメモリなどの計算資源がプログラム中でどのように使用されるかを型を用いて推論するための手法が五十嵐，小林らにより提案されている．彼らの手法では，各計算資源に対して起こりうるアクセス列の集合を使用法表現と呼ばれる式として推論する．しかしながら，推論された使用法表現が表すアクセス列の集合が，仕様として許されるアクセス列からなる言語に包含されているかどうかを判定するアルゴリズムが考案されておらず，計算資源使用法検証の完全な自動化は達成できていなかった．本論文では，ある特定の言語クラスに属する仕様に対し，そのような包含判定を行うための健全かつ完全なアルゴリズムを提案する．仕様として任意の正則言語を許す場合には，包含判定問題は決定不能なので，我々のアルゴリズムが扱う仕様のクラスは，1つの入力記号について，遷移元および遷移先の状態がたかだか1つしか存在しない有限オートマトンが受理する言語のクラスとして与えられ，正則言語のクラスより小さい．しかしながら，ファイルなど現実の計算資源の仕様の多くは，我々のアルゴリズムで扱える言語のクラスに属する．したがって，本論文のアルゴリズムを五十嵐と小林らによる従来の研究と組み合わせることにより，ファイルなど多くの計算資源の使用法検証を自動化できる．

An Algorithm to Decide Conformance of Resource Usage to Specification for Resource Usage Verification

Futoshi Iwama,[†] Atsushi Igarashi[††] and Naoki Kobayashi[†]

In our previous work, we have developed a type-based method for inferring how resources such as files and memory are accessed in a program. Due to the lack of an algorithm for deciding whether the inferred resource usage conforms to the specification, however, it was not possible to verify automatically that resources are accessed according to the specification. In this paper, we propose a sound and complete algorithm for deciding the conformance of the inferred resource usage to the specification. Since the language denoted by the inferred resource usage belongs to a class larger than the class of the context-free languages, the class of specifications that our algorithm can deal with is smaller than the class of regular languages. Fortunately, however, that language class contains many of the typical resource usage specifications used in practice. Thus, by combining our algorithm with our previous method for resource usage inference, we can automatically check whether each resource is accessed according to the specification in many cases.

1. 序 論

メモリやファイル，ネットワークなどが，仕様に従ってアクセスされること（たとえば，オープンしたファイルは必ずクローズされること）をプログラムの実行前に検証する手法がさかんに研究されている[1),2),4),6),14),15)]．

† 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
†† 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University

五十嵐，小林ら[6),8)]は，従来，計算資源（とその仕様）の種類により個別に扱われてきた上記のような検証問題（**計算資源使用法検証**の問題と呼ぶ）を統一的に扱うため，計算資源がプログラム中でどのように使用されるかを推論するための一般的な型システムを与えた．彼らの型システムでは，計算資源の型に使用順序に関する情報を付加し拡張した型を用いており，この拡張した型を推論することで，各計算資源に対して起こりうるアクセス列の集合を**使用法表現**と呼ばれる式として機械的に推論することができる．したがって，この推論された使用法表現が表すアクセス列の集合が仕様として許されているアクセス列の集合に含まれて

いることを判定することで，計算資源使用法の検証を行うことができる．

たとえば次のような名前 f1 のファイルの先頭3行を名前 f2 のファイルにコピーしている Objective Caml のプログラムを考える．

```
let ic = open_in  "f1" in
let oc = open_out "f2" in
let rec fcp n ic oc
    = if (n <= 0)
      then ()
      else let s = input_line ic
           in  output_string oc (s^"\n");
               fcp (n-1) ic oc
in fcp 3 ic oc; close_in ic; close_out oc;;
```

上のプログラムでは，ic，oc が読み込み用，書き込み用のファイルをそれぞれ指す．関数 fcp が，ic から oc へ n 行コピーする関数であり，input_line ic が ic からの1行の読み出し，output_string oc s が oc へ1行 s の書き込みを行っている関数である．最後の行では，関数 fcp を使って ic からの3行を oc にコピーし，その後，close_in ic，close_out oc で読み込み用，書き込み用のファイルを閉じている☆．

このプログラムの ic，oc に対して，彼らの型システムは，$(\mu\rho.((R;\rho)\&\mathbf{0}));C$，$(\mu\rho.((W;\rho)\&\mathbf{0}));C$ という使用法表現を推論する．ここで，R，W，C は，アクセスの種類（それぞれ読み出し，書き込み，閉じる操作を表す）を表すラベルである．使用法表現中の $\mu\rho.$ は再帰，; は逐次実行，& は非決定的な分岐，$\mathbf{0}$ は空のアクセス列を表す．直観的には，上記の使用法表現 $(\mu\rho.((R;\rho)\&\mathbf{0}));C$ は，次の文脈自由文法において ρ_1 から生成されるラベル列の集合に相当する（使用法表現の正式な定義は 2.2 節で与えられる）．

$$\rho_1 \to \rho C \qquad \rho \to R\rho \mid \epsilon$$

一方，読み込み専用ファイルの仕様は，任意回の読み込み操作の後，1回閉じなければならないというもので，おおよそ正則表現 R^*C が意味するラベル列の集合で表される（仕様の正式な定義は 2.3 節で与えられる）．よって，推論の結果得られた使用法表現 $\mu\rho.((R;\rho)\&\mathbf{0});C$ が表すラベル列の集合が仕様 R^*C に含まれていることを判定することにより，ic がプログラム中で仕様どおりにアクセスされていることを検証できる．

しかしながら，上記のような推論された使用法と仕様との間の包含関係を機械的に判定するアルゴリズムが考案されていなかったため，計算資源使用法検証の完全な自動化が達成できていなかった．

本研究では，正則言語の特定の部分クラスに属する仕様に対して，上記の包含判定を行うための健全かつ完全なアルゴリズムを提案する．このクラスは，1つの入力記号について，遷移元および遷移先の状態がたかだか1つしか存在しない有限オートマトンが受理する言語のクラスとして定義される．仕様のクラスには，メモリオブジェクトの仕様（初期化されたあと，解放される）や読み込み専用ファイルの仕様（読み出しアクセスが任意回行われた後，ちょうど1回クローズされる）など多くの計算資源の仕様が含まれる．したがって，五十嵐，小林らにより提案されている型システム[6),8)] による解析と本研究で提案する包含関係判定アルゴリズムとを組み合わせることにより，多くの計算資源の使用法検証を自動化することが可能になる．

なお，仕様のクラスを一般の正則言語のクラスとした場合，上記の包含判定問題は決定不能である．この理由は，直観的には，使用法表現が文脈自由言語どうしのシャッフルを表現できるためであるが，厳密な議論については付録 A.3 を参照されたい．

以降の構成

2章では，仕様，使用法表現について定義し，計算資源使用法検証の問題における使用法表現と仕様との間の包含関係の判定問題を定式化する．3章では，2章で定式化した，使用法表現と仕様との間の包含関係を判定するアルゴリズムについて記述する．4章では，使用法表現をさらに拡張した場合における判定手法について簡単に述べる．そして，5章で関連研究について触れ，6章でまとめと今後の課題について述べる．

2. 計算資源使用法検証における言語包含関係の判定問題

本章では，計算資源使用法検証における言語包含関係の判定問題を定式化する．2.1 節では，まず計算資源へのアクセス列を表すトレースとトレース集合の概念を導入する．続いて 2.2 節で，使用法表現の構文を定義し，その意味をトレース集合として与える．2.3 節では，本論文で計算資源の仕様として扱うトレース集合のクラスを定義する．これらの定義を用い，2.4 節で，使用法表現と仕様との間の言語包含判定の問題を

☆ 通常，計算資源へのアクセスには例外が発生する可能性がある．実際，上記の関数 input_line ic は ic からこれ以上読み込む行がない場合，例外 End_of_File を発生させる．本論文では議論を単純にするため例外については扱わないが，例外発生，例外処理があるプログラムに対しての検証も本論文のアルゴリズムを拡張することで行うことができる．

定式化する．

2.1 トレース

計算資源に対して行われる操作をアクセスラベル l で表し，アクセスラベルの集合として有限集合 $\mathcal{L}$ を仮定する．たとえば，初期化，読み込み，書き込み，閉じる操作をそれぞれアクセスラベル I，R，W，C を用いて表す．

以下に定義するトレースは，プログラムの実行過程における計算資源に対するアクセスおよびプログラム終了の履歴を表す．

定義 2.1（トレース） アクセスラベル $\mathcal{L}$ に対して定義される集合 $\mathbf{Traces}(\mathcal{L}) = \mathcal{L}^* \cup \{s\downarrow \mid s \in \mathcal{L}^*\}$ の要素 t を**トレース**と呼ぶ．ここで，$\mathcal{L}^*$ は，$\mathcal{L}$ の要素の有限列の集合を表す．

$\downarrow$ は一連のアクセスの後に，実行が正常に終了することを表す特別な記号である．以降，$\mathcal{L}^*$ の要素を s で表す．

トレース $l_1 \ldots l_k$ は，プログラムの実行中に，計算資源に対するアクセスが $l_1, \ldots, l_k$ の順で起きた履歴を表し，トレース $l_1 \ldots l_k\downarrow$ は，アクセスが $l_1, \ldots, l_k$ の順で行われた後にプログラム（あるいはその一部）の実行が正常に終了するという履歴を表す．たとえば，$IRC\downarrow$ は，資源が初期化され，1 回読まれた後に閉じられて実行が終了するという履歴を表す．

計算資源があるトレース t に従って使用されうるならば，その資源は t の前方部分列（prefix）に従っても使用されうることに注意されたい．したがって，計算資源に対して起りうるアクセス列の集合は，以下のような前方部分列に関して閉じた集合として表現される．

定義 2.2（トレース集合） $S \subseteq \mathbf{Traces}(\mathcal{L})$ に対して，S のすべての前方部分列よりなる集合を $S^{\#}$ で表すこととする．つまり，$S^{\#} = \{t \in \mathbf{Traces}(\mathcal{L}) \mid \exists t'.tt' \in S\}$ と定義する．このとき，空でない集合 S で $S^{\#} \subseteq S$ を満たすもの（すなわち前方部分列に関して閉じた）を**トレース集合**と呼びメタ変数 Φ で表す．

以降，表記を簡略化するため，(r) で正則表現 r が意味する文字列の集合をも表すこととする．たとえば，$(l_1^* l_2)$ で $\{l_2, l_1l_2, l_1l_1l_2, l_1l_1l_1l_2, \ldots\}$ を表す．最小のトレース集合は，$\{\epsilon\}$ であることに注意されたい．

2.2 計算資源の使用法表現

1 章で述べたように，五十嵐と小林[6] は，プログラム実行中に各計算資源に対して起りうるアクセス列の集合（の上限）を静的に解析し，使用法表現として表す．本節では，この使用法表現の構文を定義し，その意味をトレース集合として与える．

使用法表現の構文

使用法表現 U の構文は以下のとおり（五十嵐，小林らの型システムでは以下に $\diamond U$，$\blacklozenge U$ を加えた使用法表現を扱っているが，これらの扱いについては 4 章で述べる）．

定義 2.3（使用法表現）

$$U ::= \mathbf{0} \mid l \in \mathcal{L} \mid \rho \mid U_1; U_2 \mid U_1 \otimes U_2 \mid U_1 \& U_2 \mid \mu\rho.U$$

$\mathbf{0}$ は，アクセスが起こらないことを表す．l は，l で表されるアクセスがちょうど 1 回起きることを表す．$U_1; U_2$ は，U_1 に従うアクセスの後，U_2 に従うアクセスが起きることを表す．$U_1 \otimes U_2$ は，U_1 に従うアクセスと U_2 に従うアクセスとが，インタリーブして起きる可能性があることを示す．これは，同じ計算資源が複数のスレッドによって並行にアクセスされる状況や，逐次プログラムにおいて同じ計算資源が複数の変数を介してアクセスされる状況において，全体としてどのようなアクセスが起きるかを表すのに用いる．たとえば，関数 $f(x, y)$ が変数 x を U_1 に従ってアクセスし，y を U_2 に従ってアクセスする場合，関数呼び出し $f(r, r)$ による r に対するアクセスを $U_1 \otimes U_2$ で表す．$U_1 \& U_2$ は，U_1 または U_2 のどちらか一方に従ってアクセスされることを表す☆．これは，条件分岐文などのプログラム中でのアクセス順序を表現すのに用いられる．$\mu\rho.U$ は，$\rho = U$ を満たす ρ に従って再帰的にアクセスされることを表し，ループ命令や，再帰関数などで使用される計算資源の使用法を表すのに用いる．たとえば $\mu\rho.((R; \rho)\&C)$ は，任意回の読み込み操作 R の後，閉じられる資源の使用法を表す．

使用法表現の意味

使用法表現の意味を定義するため，トレース集合間の演算を使用する．

定義 2.4 トレース集合の間に演算 $\Phi_1; \Phi_2$，$\Phi_1 \otimes \Phi_2$ を**図 1** によって定義する．ただし，s，s_i，s_i' は $\mathcal{L}^*$ の要素であり，空列 ϵ でもありうる．

例 2.5

$\{\epsilon, R, R\downarrow\}; \{\epsilon\} = \{\epsilon, R\}$.
$\{\epsilon\}; \{\epsilon, R, R\downarrow\} = \{\epsilon\}$.
$\{\epsilon, R, R\downarrow\}; \{\epsilon, W, W\downarrow\} = \{\epsilon, R, RW, RW\downarrow\}$.
$\{\epsilon, R, R\downarrow\} \otimes \{\epsilon\} = \{\epsilon\} \otimes \{\epsilon, R, R\downarrow\} = \{\epsilon, R\}$.
$\{\epsilon, R, R\downarrow\} \otimes \{\epsilon, W, W\downarrow\}$.
$= \{\epsilon, R, W, RW, WR, RW\downarrow, WR\downarrow\}$.

以下のように，使用法表現 U の意味 $[\![U]\!]$ をトレース集合として与える．

☆ & は線型論理の additive conjunction に相当する．

$$\begin{aligned}\Phi_1;\Phi_2 &= \{s_1t_2 \mid s_1\downarrow \in \Phi_1, t_2 \in \Phi_2\} \cup \{s \mid s \in \Phi_1\}\\ \Phi_1 \otimes \Phi_2 &= \{s_1s'_1s_2s'_2\ldots s_ns'_n\downarrow \mid s_1s_2\ldots s_n\downarrow \in \Phi_1, s'_1s'_2\ldots s'_n\downarrow \in \Phi_2\}\\ &\quad \cup \{s_1s'_1s_2s'_2\ldots s_ns'_n \mid s_1s_2\ldots s_n \in \Phi_1, s'_1s'_2\ldots s'_n \in \Phi_2\}\end{aligned}$$

図 1 トレース集合間の演算 $\Phi_1;\Phi_2$，$\Phi_1 \otimes \Phi_2$
Fig. 1 Operations $\Phi_1;\Phi_2$, $\Phi_1 \otimes \Phi_2$.

定義 2.6（$[\![U]\!]$） 使用法表現 U に対し，トレース集合 $[\![U]\!]$ を以下のように定義する．

$$\begin{aligned}[\![U]\!] &= [\![U]\!]_\emptyset\\ [\![\mathbf{0}]\!]_F &= \{\epsilon, \downarrow\}\\ [\![l]\!]_F &= \{\epsilon, l, l\downarrow\}\\ [\![U_1; U_2]\!]_F &= [\![U_1]\!]_F; [\![U_2]\!]_F\\ [\![U_1 \otimes U_2]\!]_F &= [\![U_1]\!]_F \otimes [\![U_2]\!]_F\\ [\![U_1 \& U_2]\!]_F &= [\![U_1]\!]_F \cup [\![U_2]\!]_F\\ [\![\rho]\!]_F &= F(\rho)\\ [\![\mu\rho.U]\!]_F &= \mathbf{lfp}(\lambda x.[\![U]\!]_{F\{\rho\mapsto x\}})\end{aligned}$$

ここで，F は使用法表現の変数からトレース集合への写像を表す．また，$\mathbf{lfp}(f)$ は，関数 f の最小不動点（$\{\epsilon\}$ を最小元，トレース集合間の包含関係を順序とする）を表す演算子である．“;”，“$\otimes$”，“&” が単調であることにより，$\lambda x.[\![U]\!]_{F\{\rho\mapsto x\}}$ も単調であることに注意．

例 2.7

$$\begin{aligned}&[\![\mu\rho.((R;\rho)\&C)]\!]\\ &\quad = \{\epsilon, C, C\downarrow, R, RC, RC\downarrow, \ldots\} = (R^*C\downarrow)^\#\end{aligned}$$

$[\![\mu\rho.(l;\rho)]\!]$ と，生成規則 $S \to lS$ のみからなる文脈自由文法によって生成される言語とは異なることに注意されたい．前者は $[\![\epsilon, l, ll, lll, \cdots]\!] = (l^*)$ であるが，後者は $\emptyset$ である．

2.3 計算資源の仕様

仕様を表すトレース集合のクラスを，以下のような制限付きの有限状態オートマトン M の受理言語として定義する．

定義 2.8（仕様オートマトン） 仕様オートマトンとは，有限状態オートマトン $M = (Q, \mathcal{L}, \delta_M, q_s, F)$ のうち，遷移関数 δ_M（$\in Q \times \mathcal{L} \rightharpoonup Q$）が以下の条件を満たすものである．

$$\begin{aligned}&\forall q_1, q_2, q'_1, q'_2 \in Q.\forall a \in \mathcal{L}.\\ &\quad (\delta_M(q_1, a) = q_2 \wedge \delta_M(q'_1, a) = q'_2)\\ &\qquad \Rightarrow \quad (q_1 = q'_1 \wedge q_2 = q'_2)\end{aligned}$$

上の条件は，1 つの入力記号に対して遷移元，遷移先が一意に決定されることを表す．

以降の定義を簡単にするため，遷移関数 δ を拡張した $\hat{\delta}$ を次のように定める（遷移関数は部分関数であることに注意されたい）．

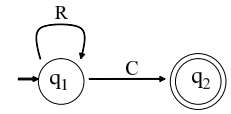
$$\hat{\delta}(q, s) \stackrel{\text{def}}{=} \begin{cases} q & \text{if } s = \epsilon\\ \hat{\delta}(\delta(q, l), s') & \text{if } s = ls'\end{cases}$$

定義 2.9（計算資源の仕様） 仕様オートマトン M が与えられたとき，M が表す仕様 $Spec(M)$ を以下によって定義する．

$$Spec(M) = \{s\downarrow \mid s \in Lang(M)\}^\# \cup \{\epsilon\}$$

ここで，$Lang(M)$ は，オートマトン M の受理集合を表す（$Lang(M) \stackrel{\text{def}}{=} \{s \mid \hat{\delta}(q_s, s) \in F\} \subseteq \mathcal{L}^*$）．

例 2.10 次のような仕様オートマトン M_{rof} を考える．

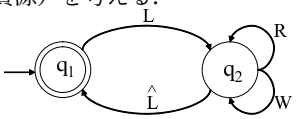


ここで，M_{rof} の開始状態は q_1，受理状態は q_2 であり，

$$\begin{aligned}Spec(M_{rof}) &= (R^*C\downarrow)^\#\\ &= \left\{\begin{array}{l}\epsilon, C, C\downarrow, R, RC, RC\downarrow,\\ RR, RRC, RRC\downarrow, \ldots\end{array}\right\}\end{aligned}$$

である． □

例 2.11 次のような仕様オートマトン M_{lrw} で表される仕様を持つ資源（ロックを獲得中のみ読み書きできる計算資源）を考える．



ただし，ラベル L，$\hat{L}$ は，ロックの獲得，解放を表すものとする．このとき，

$$Spec(M_{lrw}) = ((L(R+W)^*\hat{L})^*\downarrow)^\#$$

である． □

1 章でも触れたように，仕様オートマトンとして制限のないオートマトン M_{gen} を認めると，$[\![U]\!] \subseteq Spec(M_{gen})$ は決定不能となる．証明の詳細は付録 A.3 を参照されたい．

2.4 使用法表現と仕様との間の包含関係判定問題

本論文で扱う問題は次のように定式化される．

「仕様オートマトン M と使用法表現 U を与えられたとき，$[\![U]\!] \subseteq Spec(M)$ が成り立つか否かを判定する問題」．

例 2.12 ある計算資源 r の仕様が，例 2.10 の仕様オートマトン M_{rof} で与えられ，あるプログラム中での r の使用法が五十嵐と小林の型システム[6]によって $\mu\rho.((R;\rho)\&C)$ と推論されたとする．その場合，計算資源 r の使用法の正しさの十分条件は，$[\![\mu\rho.((R;\rho)\&C)]\!] \subseteq Spec(M_{rof})$ が成立するか否かという，上記の判定問題に帰着される．例 2.7 より，これは成り立つ．

3. 使用法表現と仕様との間の包含関係判定アルゴリズム

本章において，前章で定式化した使用法表現と仕様との間の言語包含関係を判定するためのアルゴリズムを与える．

まず，3.1 節でアルゴリズムのアイデアを述べ，3.2, 3.3 節でそのアイデアを形式的に記述する．

3.1 包含関係判定アルゴリズムのアイデア

$[\![U]\!] \subseteq Spec(M)$ を判定するあたり，一般に $[\![U]\!]$ は無限集合であるため要素のすべてを列挙しつくすことはできない．また，U はシャッフルに相当する構成子を含むので，文脈自由言語と正則言語の包含判定の手法を用いることはできない．

しかしながら，M を固定した場合，$[\![U]\!]$ の要素の中のある種のラベル列は同値と見なすことができる．たとえば，M として，例 2.10 の仕様オートマトン M_{rof} を考える．すると，ラベル列 C，RC，RRC，... は同一と見なすことができる．なぜなら，これらのラベル列はすべて M_{rof} の状態 q_1 から状態 q_2 への遷移に相当し，他のどのラベル列と ; や $\otimes$ によって結合しても，その結果はまた M_{rof} 上の同一の遷移に対応するからである．たとえば，任意の $s \in \{C, RC, RRC, \ldots\}$ について $R;s$ は正当なアクセス列を表し，一方，$C;s$ はいずれも不当なアクセス列となる．

上記のような同値関係を $\sim$ とすると，$\mathbf{Traces}(\mathcal{L})$ の $\sim$ による商集合 $\mathbf{Traces}(\mathcal{L})/\sim$ は，有限個の同値類からなる集合となる．したがって，$[\![U]\!]$ を直接計算する代わりに，$[\![U]\!]$ に含まれる元の同値類の集合 $[\![U]\!]/\sim$ を計算し，その各々の同値類が仕様に含まれていることを確かめればよい．

たとえば，仕様オートマトン M_{rof} が与えられたとき，$\mathbf{Traces}(\mathcal{L})/\sim$ は次のような集合となる．

$$\{(\epsilon),(R^*),(R^*C),\mathbf{err},(\downarrow),(R^*\downarrow),(R^*C\downarrow),\mathbf{err}\downarrow\}$$

ここで，$\mathbf{err} = \{s \mid \forall q \in Q.\hat{\delta}(q,s)$ が未定義 $\}$ である．

3.2 トレースの同値関係と $[\![U]\!]$ の同値類

本節では，前節のアイデアを形式的に記述する．

定義 3.1（トレースの同値関係） 仕様オートマトン $M = (Q, \mathcal{L}, \delta, q_s, F)$ が与えられたとき，トレースの間の同値関係 $\sim_M$ を次で定める．

- $s_1 \sim_M s_2 \stackrel{\text{def}}{=}$
$$\forall q_1, q_2 \in Q.(\hat{\delta}(q_1, s_1) = q_2 \Leftrightarrow \hat{\delta}(q_1, s_2) = q_2)$$
$$\wedge\ \forall q \in Q. \left(\begin{array}{l} reachable(q, s_1) = \{q\} \\ \Leftrightarrow reachable(q, s_2) = \{q\} \end{array} \right)$$
- $s_1\downarrow \sim_M s_2\downarrow \stackrel{\text{def}}{=} s_1 \sim_M s_2$

ここで，$reachable(q,s)$ は，q から入力 s で到達可能な状態の集合 $\{q' \mid \exists s_1, s_2.(\hat{\delta}(q, s_1) = q' \wedge s = s_1 s_2)\}$ を表す．

以後，M が文脈から明らかな場合，M を略す．

$s_1 \sim_M s_2$ の最初の条件により，演算 ";" に関して同値類が保存される．つまり $t_1 \sim t_1'$ かつ $t_2 \sim t_2'$ ならば $\{t_1\};\{t_1'\} \sim \{t_2\};\{t_2'\}$ が成り立つ．

しかしながら，この 1 番目の条件だけではトレースの同値関係として不十分である．たとえば，例 2.11 の仕様オートマトン M_{lrw} を考える．$\hat{L}L$，WR はともに，状態 q_2 を q_2 に遷移させるという点では区別できない．しかし，W との間で $\otimes$ の演算を行うと，$\{WR\} \otimes \{W\} = \{WWR, WWR, WRW\}$，$\{\hat{L}L\} \otimes \{W\} = \{W\hat{L}L, \hat{L}WL, \hat{L}LW\}$ となり，異なった遷移に対応するラベル列の集合となる．ここで，$\{WWR, WWR, WRW\}$ の要素のラベル列は M_{lrw} 上の状態 q_2 を q_2 に遷移させる正当なアクセス列であるが，$\{W\hat{L}L, \hat{L}WL, \hat{L}LW\}$ に含まれる $\hat{L}WL$ は仕様（の一部）として許されていない不当なアクセス列である．

上記の理由により，同一の状態 q 間の遷移に対応するトレースについて，他の状態を経由するものとそうでないものを区別するため，2 番目の条件を含めている．

例 3.2 例 2.11 の仕様オートマトン M_{lrw} を考える．

$L \sim LR \sim LRW$,
$R \sim W \sim RWR \sim RRWW$,
$\hat{L}L \sim R\hat{L}L \sim RW\hat{L}LWR$,
$L\hat{L} \sim LR\hat{L} \sim LRRW\hat{L}$

が成り立つ．

補題 3.3 $\sim_M$ は同値関係である．

証明 定義より明らか． □

トレースの同値関係に基づき，以下のように同値類を定める．

定義 3.4（トレース同値関係による同値類）

$$[t]_{\sim_M} \stackrel{\text{def}}{=} \{t' \mid t \sim_M t'\}$$
$$[\Phi]_{\sim_M} \stackrel{\text{def}}{=} \{t' \mid t \sim_M t', t \in \Phi\}$$

例 3.5 例 2.10 の仕様オートマトン M_{rof} を考える．

$[R]_{\sim} = (R^*) = \{R, RR, RRR, \ldots\}$

$[C]_{\sim} = (R^*C) = \{C, RC, RRC, \ldots\}$

$[C\downarrow]_{\sim} = (R^*C\downarrow) = \{C\downarrow, RC\downarrow, RRC\downarrow, \ldots\}$

が成り立つ．

この同値関係について以下が成り立つ．

補題 3.6 $t \in Spec(M) \Leftrightarrow [t]_{\sim_M} \subseteq Spec(M)$

証明 同値関係の定義より明らか． □

3.1 節で述べたように，$\mathbf{Traces}(\mathcal{L})$ は有限個の同値類に類別される．

補題 3.7 任意の仕様オートマトン M について，

$\mathbf{Traces}(\mathcal{L})/\sim_M \stackrel{\text{def}}{=} \{[t]_{\sim_M} \mid t \in \mathbf{Traces}(\mathcal{L})\}$

は有限集合である．

証明 $M = (Q, \mathcal{L}, \delta, q_s, F)$ が与えられたとき，$s \in \mathcal{L}^*$ に対して，次の集合を考える．

$$Q^M_{loop}(s) \stackrel{\text{def}}{=} \{q \mid reachable(q, s) = \{q\}\}$$
$$Q^M_{path}(s) \stackrel{\text{def}}{=} \{(q_1, q_2) \mid \hat{\delta}(q_1, s) = q_2\}$$

文脈から明らかな場合，M を略する．

$\sim$ の定義より，$s_1 \sim s_2 \Leftrightarrow (Q_{loop}(s_1) = Q_{loop}(s_2) \wedge Q_{path}(s_1) = Q_{path}(s_2))$ がいえる．よって，$\{[s]_{\sim} \mid s \in \mathcal{L}^*\}$ のサイズと $S_1 = \{(Q_{loop}(s), Q_{path}(s)) \mid s \in \mathcal{L}^*\}$ のサイズは等しく，$\mathbf{Traces}(\mathcal{L})/\sim$ のサイズはその 2 倍である．ここで $S_1 \subseteq \mathcal{P}(Q) \times \mathcal{P}(Q \times Q)$ であり，Q が有限より，S_1 は有限である． □

例 3.8 例 2.10 の仕様オートマトン M_{rof} を考える．

$$\mathbf{Traces}(\mathcal{L})/\sim_{M_{rof}} = \left\{ \begin{array}{l} [\epsilon]_{\sim}, [\downarrow]_{\sim}, [R]_{\sim}, [R\downarrow]_{\sim}, \\ [C]_{\sim}, [C\downarrow]_{\sim}, [W]_{\sim}, [W\downarrow]_{\sim} \end{array} \right\}$$

である．W は R，C 以外のラベルであればよい．

トレースの集合の間に以下のような同値関係を定める．

定義 3.9（トレースの集合間の同値関係） 仕様オートマトン M に対して，

$Err(M) \stackrel{\text{def}}{=} \{s | Q^M_{path}(s) = \emptyset\} \cup \{s\downarrow | Q^M_{path}(s) = \emptyset\}$

を定義する．このとき，トレースの集合の間の関係 $S_1 \cong_M S_2$ を次の何れかを満たす関係として定義する．

- $S_1 = S_2$
- $S_1 \cap Err(M) \neq \emptyset$ かつ $S_2 \cap Err(M) \neq \emptyset$

$S_1 \cong_M S_2$ は同値関係である．M が文脈から明らかなときは省略して $\cong$ と書く．

上記の定義において，集合 $Err(M)$ は，不当なアクセス列からなる集合を表す．$S_1 \cong_M S_2$ の定義の 2 番目の条件は，不当なアクセス列を含むトレース集合を（必ず仕様に違反するという点で）すべて同一視することを意味する．

上の観察より，以下の事実が容易に導かれる．

補題 3.10 $S_1 \cong_M S_2$ ならば，$S_1 \subseteq Spec(M) \Leftrightarrow S_2 \subseteq Spec(M)$.

補題 3.6 と上記の補題により，$[\![U]\!] \subseteq Spec(M)$ を判定するためには，$\alpha_M(U) \cong_M [[\![U]\!]]_{\sim_M}$ を満たす $\alpha_M(U)$ を求め，$\alpha_M(U) \subseteq Spec(M)$ が成り立つか否かを判定すればよいことが分かる．以下ではそのような $\alpha_M(U)$ の構成法を与える．

$\alpha_M(U)$ の構成の前に，演算 “;”，“$\otimes$” は $\sim_M$ による同値類を（$\cong_M$ の下で）保存することを確認する．

補題 3.11 トレース集合の演算 “;”，“$\otimes$” について以下が成り立つ．

(1) $[[t_1]_{\sim_M}; [t_2]_{\sim_M}]_{\sim_M} \cong_M [\{t_1\}; \{t_2\}]_{\sim_M}$

(2) $[[t_1]_{\sim_M} \otimes [t_2]_{\sim_M}]_{\sim_M} \cong_M [\{t_1\} \otimes \{t_2\}]_{\sim_M}$

証明 付録 A.1 で証明する．

上の補題より，$\alpha_M(U)$ を以下のように定義できる．

定義 3.12 使用法表現の変数 ρ から，トレース集合への写像を F とおくとき，U の M に対する抽象化関数 $\alpha_M(U)$ を次で定義する．

$$\begin{aligned}
\alpha_M(U) &= \alpha^{\emptyset}_M(U) \\
\alpha^F_M(\mathbf{0}) &= [\{\epsilon, \downarrow\}]_{\sim_M} \\
\alpha^F_M(l) &= [\{\epsilon, l, l\downarrow\}]_{\sim_M} \\
\alpha^F_M(U_1; U_2) &= [\alpha^F_M(U_1); \alpha^F_M(U_2)]_{\sim_M} \\
\alpha^F_M(U_1 \otimes U_2) &= [\alpha^F_M(U_1) \otimes \alpha^F_M(U_2)]_{\sim_M} \\
\alpha^F_M(U_1 \& U_2) &= [\alpha^F_M(U_1) \cup \alpha^F_M(U_2)]_{\sim_M} \\
\alpha^F_M(\rho) &= [F(\rho)]_{\sim_M} \\
\alpha^F_M(\mu\rho.U) &= \mathbf{lfp}(\lambda x.\alpha^{F\{\rho \mapsto x\}}_M(U))
\end{aligned}$$

$\mathbf{lfp}(f)$ は，f の最小不動点（$\{\epsilon\}$ を最小元，集合の包含関係を順序とする）を意味する．

$\alpha_M(U)$ が $[\![U]\!]$ と比べて異なるのは，各演算のたびに，$\sim_M$ による同値類をとるという点である．また，$\alpha_M(U)$ が単調であり，同値類の数が有限（補題 3.7）であることから，$\alpha_M(U)$ を有限回のステップで計算できる．以降，文脈から明らかな場合，M を略して $\alpha(U)$ と書く．

例 3.13 例 2.10 の M_{rof} を考える．

$$\begin{aligned}
&\alpha_{M_{rof}}(R; R; C) \\
&\quad = [[([\{\epsilon, R, R\downarrow\}]_{\sim}; [\{\epsilon, R, R\downarrow\}]_{\sim})]_{\sim} \\
&\qquad ; [\{\epsilon, C, C\downarrow\}]_{\sim}]_{\sim} \\
&\quad = [[\{\epsilon, R, R\downarrow\}]_{\sim}; [\{\epsilon, C, C\downarrow\}]_{\sim}]_{\sim} \\
&\quad = [\{\epsilon, R, RC, RC\downarrow\}]_{\sim} = [\{\epsilon, C, C\downarrow\}]_{\sim}
\end{aligned}$$

である．

例 3.14 例 2.11 の M_{lrw} を考える．

$$\begin{aligned}
&\alpha_{M_{lrw}}(L; \mu\rho.(((R;W)\otimes\rho)\&\mathbf{0}); \hat{L})\\
&\quad = [[\{\epsilon, \downarrow, L, L\downarrow\}]_\sim\\
&\qquad ; [[\{\epsilon, \downarrow, R, R\downarrow\}]_\sim; [\{\epsilon, \hat{L}, \hat{L}\downarrow\}]_\sim]_\sim]_\sim\\
&\quad = [[\{\epsilon, L, L\downarrow\}]_\sim\\
&\qquad ; [\{\epsilon, R, \hat{L}, \hat{L}\downarrow, R\hat{L}, R\hat{L}\downarrow\}]_\sim]_\sim\\
&\quad = [\{\epsilon, L, LR, L\hat{L}, L\hat{L}\downarrow, LR\hat{L}, LR\hat{L}\downarrow\}]_\sim\\
&\quad = [\{\epsilon, L, L\hat{L}, L\hat{L}\downarrow\}]_\sim
\end{aligned}$$

である．

補題 3.15 $\alpha_M(U) \cong_M [[U]]_{\sim_M}$

証明 付録 A.2 で証明．

以上より，$[\![U]\!]$ と $Spec(M)$ との間の包含判定を行うには，$[\![U]\!]$ の同値類 $\alpha_M(U)$ を計算した後，$\alpha_M(U)$ と $Spec(M)$ との間の包含関係を判定すればよいことが分かる．

系 3.16（抽象化関数の健全性と完全性）

$[\![U]\!] \subseteq Spec(M) \Leftrightarrow \alpha_M(U) \subseteq Spec(M)$

証明 補題 3.6, 3.10 および補題 3.15 より明らか．□

3.3 包含関係判定アルゴリズム

前節の $\alpha_M(U)$ は無限集合であるため，実際に $\alpha_M(U) \subseteq Spec(M)$ の判定を行うためには，$\alpha_M(U)$ そのものを計算する代わりに，$\alpha_M(U)$ に含まれる同値類を列挙し，それらが $Spec(M)$ に包含されることを判定する．すなわち，$\alpha_M(U)/\sim_M \subseteq Spec(M)/\sim_M$ の判定を行う．

以下，包含関係判定の具体的なアルゴリズムを記述する．

まず，同値関係 $\sim_M$ による同値類の有限の表記法を定める．

$M = (Q, \mathcal{L}, \delta_M, q_s, F)$ とおくとき，$s \in \mathcal{L}^*$ に対して集合 $Q_{loop}(s)$，$Q_{path}(s)$（補題 3.7 の証明で定義）を考える．このとき，仕様オートマトンの制限 **UL** より，次のいずれか 1 つが成り立つ．

(a) $Q_{loop}(s) = Q$

(b) $\exists\, q \in Q.(Q_{loop}(s) = \{q\})$

(c) $\exists\, q_1, q_2 \in Q.\left(\begin{array}{l} Q_{path}(s) = \{(q_1, q_2)\} \\ \wedge\, Q_{loop}(s) \neq \{q_1\} \end{array}\right)$

(d) $Q_{path}(s) = \emptyset$

これに基づき $\sim_M$ による同値類を以下の表現（同値類表現と呼ぶ）u で表す．

$$\begin{aligned}
u &::= pe\\
p &::= \epsilon \mid q \mid q_1q_2 \mid \mathbf{err}\\
e &::= \epsilon \mid \downarrow
\end{aligned}$$

$p = q$, q_1, q_1q_2, **err** が (a), (b), (c), (d) の場合に対応している．たとえば，s が $Q_{path}(s) = (q_1', q_2')$ かつ $Q_{loop}(s) \neq \{q_1'\}$ であるとき同値類 $[s\downarrow]_{\sim_M}$ を $q_1'q_2'\downarrow$ と表記する．

同値類表現 u が意味する同値類 $[\![u]\!]_M$ は以下のようになる．

$$\begin{aligned}
&[\![\epsilon]\!]_M = \{\epsilon\}\\
&[\![q]\!]_M = \{l \mid \delta_M(q,l) = q\}^+\\
&[\![q_1q_2]\!]_M = \left\{ s \;\middle|\; \begin{array}{l} \hat{\delta}_M(q_1, s) = q_2, \\ s \notin ([\![\epsilon]\!]_M \cup [\![q_1]\!]_M) \end{array} \right\}\\
&[\![\mathbf{err}]\!]_M = \{s \mid \forall q \in Q.\hat{\delta}(q,s)\} \text{ が未定義}\\
&[\![p\downarrow]\!]_M = [\![p]\!]_M\downarrow
\end{aligned}$$

ここで，言語 L に対して，$L\downarrow = \{s\downarrow \mid s \in L\}$ と定義する．

この同値類表現を使用し，以下の 2 つのステップで包含関係 $[\![U]\!] \subseteq Spec(M)$ の判定を行うことができる．

アルゴリズム

Step 1：$\alpha_M(U)/\sim_M = \{u_1, \ldots, u_n\}$ を計算

Step 2：各 $u_i \in \alpha_M(U)/\sim_M$ について $[\![u_i]\!]_M \subseteq Spec(M)$ を判定 □

Step 1 を実行するアルゴリズムは定義 3.12 の関数 $\alpha(U)$ を使い，次のように構築できる．

アルゴリズム（$\alpha_M(U)/\sim_M$ の計算）

$M = (Q, \mathcal{L}, q_s, \delta_M, Q_F)$ のとき，同値類の集合 $\alpha_M(U)/\sim_M$ は以下の関数 $abst(M, U)$ で計算される：

$$\begin{aligned}
&abst(M, U) = ab_M(\emptyset, U)\\
&ab_M(F, \mathbf{0}) = \{\downarrow\}\\
&ab_M(F, l) = \begin{cases} \{q_1q_2\downarrow\} & \text{if } \delta_M(q_1, a) = q_2 \\ & \text{and } q_1 \neq q_2 \\ \{q_1\downarrow\} & \text{if } \delta_M(q_1, a) = q_1 \\ \{\mathbf{err}\} & \text{otherwise} \end{cases}\\
&ab_M(F, U_1; U_2) = ab_M(F, U_1); ab_M(F, U_2)\\
&ab_M(F, U_1 \otimes U_2) = ab_M(F, U_1) \otimes ab_M(F, U_2)\\
&ab_M(F, U_1 \& U_2) = ab_M(F, U_1) \& ab_M(F, U_2)\\
&ab_M(F, \rho) = F(\rho)\\
&ab_M(F, \mu\rho.U) = lfp_M(F\{\rho \mapsto \{\epsilon\}\}, \rho, U)
\end{aligned}$$

$$\begin{aligned}
&lfp_M(F, \rho, U) =\\
&\quad \text{let} \quad x = ab_M(F, U)\\
&\quad \text{in} \quad \text{if } (x = F(\rho)) \text{ then } x\\
&\qquad\quad\;\; \text{else } lfp_M(F\{\rho \mapsto x\}, \rho, U)
\end{aligned}$$

ただし，F は使用法表現の変数から同値類表現の集合への対応であり，同値類表現の集合の間の演算 ";", "⊗", "&" は**図 2** で定義される． □

Step 2 の $[\![u]\!]_M \subseteq Spec(M)$ の判定は，$u \in Spec(M)/\sim_M$ で行うことができる．ここで

$$
\begin{aligned}
V_1;V_2 &= \{(p_1;p_2)e_2 \mid p_1\downarrow \in V_1,\ p_2e_2 \in V_2\} \cup \{p_1 \mid p_1 \in V_1\}\\
V_1 \otimes V_2 &= \{(p_1 \otimes p_2)(e_1 \otimes e_2) \mid p_1e_1 \in V_1,\ p_2e_2 \in V_2\}\\
V_1\&V_2 &= V_1 \cup V_2
\end{aligned}
$$

$$
(p_1;p_2) = \begin{cases} p_2 & \text{if } p_1 = \epsilon \\ p_1 & \text{if } p_2 = \epsilon \\ q & \text{if } p_1 = p_2 = q \\ q_1q_2 & \text{if } p_1 = q_1 \text{ and } p_2 = q_1q_2 \\ q_1q_2 & \text{if } p_1 = q_1q_2 \text{ and } p_2 = q_2 \\ q_1q_3 & \text{if } p_1 = q_1q_2 \text{ and } p_2 = q_2q_3 \\ \mathbf{err} & \text{otherwise} \end{cases}
\qquad
(p_1 \otimes p_2) = \begin{cases} p_2 & \text{if } p_1 = \epsilon \\ p_1 & \text{if } p_2 = \epsilon \\ q & \text{if } p_1 = p_2 = q \\ \mathbf{err} & \text{otherwise} \end{cases}
$$

$$
e_1 \otimes e_2 = \begin{cases} \epsilon & \text{if } e_1 = \epsilon \text{ or } e_2 = \epsilon \\ \downarrow & \text{if } e_1 = e_2 = \downarrow \end{cases}
$$

図 2 $abst(M, U)$ で使用される $V_1;V_2$, $V_1 \otimes V_2$, $V_1\&V_2$ の定義
Fig. 2 Operations $V_1;V_2$, $V_1 \otimes V_2$, $V_1\&V_2$ used in function $abst(M, U)$.

$$
\begin{aligned}
&Spec(M)/\sim_M\\
&\quad = \{\epsilon, q_s\} \cup \{q_sq \mid q \in Q\} \cup\\
&\qquad \{\downarrow, q_s\downarrow \mid q_s \in F\} \cup \{q_sq\downarrow \mid q \in F\}
\end{aligned}
$$

である．

例 3.17 例 2.10 の M_{rof} を考える．
M_{rof} で仕様が与えられている計算資源の使用法表現が $\mu\rho.((R;\rho)\&C)$ と推論されたときの問題 $[\![\mu\rho.((R;\rho)\&C)]\!] \subseteq Spec(M_{rof})$ は次のように判定される．
$abst(M_{rof}, \mu\rho.((R;\rho)\&C))$ の計算は，

$$
\begin{aligned}
&ab_{M_{rof}}(\{\rho \mapsto \{\epsilon\}\}, ((R;\rho)\&C)) = \{q_1, q_1q_2\downarrow\}\\
&ab_{M_{rof}}(\{\rho \mapsto \{q_1, q_1q_2\downarrow\}\}, ((R;\rho)\&C))\\
&\quad = \{q_1, q_1q_2\downarrow\}
\end{aligned}
$$

より，$abst(M_{rof}, \mu\rho.((R;\rho)\&C)) = \{q_1, q_1q_2\downarrow\}$.
ここで

$$
Spec(M_{rof})/\sim_{M_{rof}} = \{\epsilon, q_1, q_1q_2, q_1q_2\downarrow\}
$$

より，$[\![q_1]\!]_{M_{rof}}$，$[\![q_1q_2\downarrow]\!]_{M_{rof}} \subseteq Spec(M_{rof})$ が成り立ち，よって，$[\![\mu\rho.((R;\rho)\&C)]\!] \subseteq Spec(M_{rof})$ が成り立つことが分かる．

アルゴリズムの計算量

アルゴリズムの時間計算量を簡単に議論する．U のサイズは一般にプログラムのサイズに比例し，仕様オートマトンの状態数は定数である．そこで，U のサイズを $|U|$ とし上記アルゴリズムの計算量を評価する．

Step 1 において関数 $ab_M(F, U)$ の計算量は，上記のアルゴリズムのとおり計算すると $O(c^{|U|})$ となる（c は定数）．しかし，実際は次のように，より効率良く計算を行うことができる．$abst(M, U)$ は次の形の連立不等式の最小解となる．

$$
\begin{aligned}
&x_1 \supseteq ab_M(F, U_1)\\
&\quad \cdots\\
&x_n \supseteq ab_M(F, U_n)\\
&F = \{\rho \mapsto x, \rho_1 \mapsto x_1, \ldots, \rho_n \mapsto x_n\}
\end{aligned}
$$

ここで $U_1, \ldots, U_n$ は $\mathbf{0}$, l, ρ_i, $\rho_i;\rho_j$, $\rho_i \otimes \rho_j$, $\rho_i\&\rho_j$ のいずれかの形である．

$$
\begin{aligned}
&x_i^{(0)} = ab_M(F^{(0)}, U_1) \ \ (i = 1, \ldots, n)\\
&F^{(0)} = \{\rho \mapsto \{\epsilon\}, \rho_1 \mapsto \{\epsilon\}, \ldots, \rho_n \mapsto \{\epsilon\}\}\\
&x_i^{(k+1)} = ab_M(F^{(k)}, U_i) \ \ (i = 1, \ldots, n)\\
&F^{(k+1)} = \{\rho \mapsto x^{(k)}, \rho_1 \mapsto x_1^{(k)}, \ldots, \rho_n \mapsto x_n^{(k)}\}
\end{aligned}
$$

を考えるとき，$x^{(m)} = x^{(m+1)}$，$x_i^{(m)} = x_i^{(m+1)}$ $(i = 1, \ldots, n)$ を満たす最小の m に対して $ab_M(\{\epsilon\}, U) = x_1^{(m)}$ となる．

例 3.18 $U = L;(\mu\rho.((R;\rho)\&W));\hat{L}$ の場合，$U = abst(M, U)$ は連立不等式

$$
\begin{aligned}
&x_1 \supseteq ab_M(F, L;\rho_2)\\
&x_2 \supseteq ab_M(F, \rho_3;\rho_6)\\
&x_3 \supseteq ab_M(F, \rho_4\&\rho_5)\\
&x_4 \supseteq ab_M(F, R;\rho_3)\\
&x_5 \supseteq ab_M(F, W)\\
&x_6 \supseteq ab_M(F, \hat{L})\\
&F = \{\rho_1 \mapsto x_1, \ldots, \rho_6 \mapsto x_6\}
\end{aligned}
$$

の最小解となる． □

このように不等式に展開して計算を行う場合，繰返し回数 m は $O(|U|)$ である．不等式の個数 n は $O(|U|)$ であり，個々の $ab_M(F, U_i)$ の計算量は $O(1)$ なので，繰返し 1 回あたりの計算量は $O(|U|)$ である．したがって，$ab_M(F, U)$ の計算量は $O(|U|^2)$ である．

4. 使用法表現の拡張

五十嵐と小林[6] が導入した使用法表現には前章で我々が扱わなかった以下の構成子が含まれている．本章では，それらの拡張された使用法表現に対しての判定手法について概略のみ述べる．定義の詳細および証明は，文献 7) を参照されたい．

定義 4.1（拡張された使用法表現）

$U \ ::= \ \cdots \mid \Diamond U \mid \blacklozenge U$

$\Diamond U$ は，U に従ったアクセスが遅延され，対象となる部分プログラムの評価時ではなく，評価後（評価の結果得られた値が使用される時点）に起きるかもしれないことを表す．たとえば，関数 $\lambda z.\mathbf{read}(x)$ の中での計算資源 x の使用法を $\Diamond R$ で表す（$\mathbf{read}(x)$ は x に対する読み込みを行う命令とする）．ここで，x へのアクセスは，$\lambda z.\mathbf{read}(x)$ の評価時ではなく，$\lambda z.\mathbf{read}(x)$ の適用時に起きることに注意されたい．$\blacklozenge U$ は，U 中の $\Diamond$ をキャンセルし，U に従ったアクセスが，対象となる部分プログラムの評価時に必ず起きることを表す．たとえば，$(\lambda z.\mathbf{read}(x))()$ 中の x の使用法は，$\blacklozenge\Diamond R$ で表される．

使用法表現 $(\Diamond(l_1; l_2)); l_3$ は，$l_1; l_2$ によって表されるアクセスが遅延され，l_3 によって表されるアクセスとインタリーブして起きるかもしれないことを表す．したがって，$(\Diamond(l_1; l_2)); l_3$ が表すアクセス列の集合は $\{l_1 l_2 l_3\downarrow, l_1 l_3 l_2\downarrow, l_3 l_1 l_2\downarrow\}^{\#}$ である．一方，使用法表現 $(\blacklozenge\Diamond(l_1; l_2)); l_3$ は $\{l_1 l_2 l_3\downarrow\}^{\#}$ を表す．

拡張された使用法表現が意味する言語

$\Diamond U$，$\blacklozenge U$ を含んだ使用法表現の意味は次の拡張トレースの集合で与えられる．

定義 4.2（拡張トレース）

拡張トレースの集合 $\mathbf{ETraces}(\mathcal{L})$ を以下で定める．

$$\mathbf{ETraces}(\mathcal{L}) \stackrel{\text{def}}{=} \mathbf{Traces}(\mathcal{L}) \cup \{(s,t) | s \in \mathcal{L}^*, t \in \mathbf{Traces}(\mathcal{L})\}$$

$\mathbf{ETraces}(\mathcal{L})$ の要素を $\hat{t}$ で表す．

拡張トレース (s,t) は，アクセス列 st のうち，t で表されるアクセスが遅延されるかもしれないことを表す．使用法表現の $\Diamond$ がついた部分が拡張トレース (s,t) 中の t の部分に対応する．

拡張された使用法表現 U の意味 $[\![U]\!]$ は，上で定義した拡張トレースの集合として与えられる．たとえば，$[\![R; \Diamond W]\!]$ は，$\{\epsilon, R, (R,W), (R, W\downarrow)\}$ である，計算機資源使用法の正しさの十分条件は，包含関係 $[\![\blacklozenge U]\!] \subseteq \mathit{Spec}(M)$ で与えられる（$[\![\blacklozenge U]\!]$ は通常のトレースのみからなる集合であることに注意）．

拡張使用法表現と仕様との間の包含関係の判定

包含関係 $[\![\blacklozenge U]\!] \subseteq \mathit{Spec}(M)$ を判定するためには，拡張トレースの間に 3.2 節と同様な同値関係を定めてやればよい．

定義 4.3 仕様オートマトン $M = (Q, \mathcal{L}, \delta, q_s, F)$ が与えられたとき，拡張トレースの間の同値関係 $\approx_M$ を次で定める．

- $t_1 \approx_M t_2 \stackrel{\text{def}}{=} t_1 \sim_M t_2$
- $(s_1, t_1) \approx_M (s_2, t_2) \stackrel{\text{def}}{=} s_1 \sim_M s_2 \ \wedge \ t_1 \sim_M t_2$

前章と同様に，$\mathbf{ETraces}(\mathcal{L})/\approx$ は有限集合となるので，$[\![\blacklozenge U]\!] \subseteq \mathit{Spec}(M)$ を判定するためには，$[\![\blacklozenge U]\!]/\approx$ を計算し，それが $\mathit{Spec}(M)/\approx$ に包含されるか否かを判定すればよい．

5. 関 連 研 究

近年，プログラムの種々の性質の検証のために，言語の包含関係判定の問題に関する研究がいくつか行われている[9)~12)]．

Minamide[11] は，PHP プログラムによって動的に生成される HTML 文章が適格なものであることを，プログラムの静的な解析により検証している．その研究では，生成されうる文字列の集合の近似を文脈自由文法として求め，生成される HTML 文書木の高さを制限することによって，検証問題を文脈自由言語と正則言語の包含関係の判定に帰着している．また，Minamide と Tozawa[12] は，上記の枠組みにおける文書木の高さの制限を取り除くために，文脈自由言語と制限のついた文脈自由文法（regular hedge grammar）によって生成される言語の間の包含関係の判定アルゴリズムを与えた．

Kobayashi ら[10] は，並行計算モデルである π 計算に計算資源へのアクセスプリミティブを加えた体系に対する計算資源使用法の検証問題を，本研究と同様，ある種の使用法表現と仕様との間の包含関係の判定問題に帰着している．その研究で用いられている使用法表現は我々のものより制限されており（具体的には，我々の使用法表現と比べて，逐次実行を表す構成子 $U_1; U_2$ がない），ペトリネット言語（ペトリネットにより受理されるラベル列の集合）の部分クラスしか表現できない．そのため，その研究においては仕様として任意の正則言語を扱うことができる．

Iwama と Kobayashi[9] は，JAVA バイトコード上でのロックの獲得命令，解放命令の整合性を検証する問題を扱っている．彼らの手法では，ロックの獲得および解放の列を表す使用法表現を推論し，その使用法表現が表す言語と仕様との包含判定問題に帰着する．

ロックの整合性の仕様（ロックの獲得と同じ回数だけロックの解放が起きるという性質）は，正則言語では表すことができないが，包含関係判定のための健全かつ完全なアルゴリズムを与えている．これは，彼らの使用法表現が本研究のそれよりも制限されていること，および仕様が 1 つに定まっていることにより可能となっている．

計算資源使用法の統一的な検証という観点からの関連研究として，型状態（typestate）を用いた検証手法があげられる[3),13),16),17)]．型状態の枠組みでは，計算資源の仕様を有限状態オートマトンとして表し，計算資源の型にそのオートマトンの状態を付加することにより，各アクセスが正しい状態のときにのみ起きることを検証する．型状態の枠組みでは，五十嵐と小林の枠組みと異なり，正則言語で表される任意の仕様を扱うことができるが，これは，$\otimes$ に相当する構成子を用いないためである．一方，その代償として，並行プログラムを扱えないこと，alias の可能性がある（同一の計算資源が複数の変数から指されている）場合に，すべての alias のパターンについて，個別に型の解析をしなければならないこと，などの欠点がある．後者については，たとえば，2 引数の関数 $f(x, y)$ を解析する場合，x と y が同じ計算資源に束縛されている場合とそうでない場合とで，2 重に解析をしなければならない．したがって，引数の数に対して最悪指数オーダの計算コストがかかる．また，x が計算資源のリストである場合には，すべての alias のパターンをつくすのは不可能である．

6. 結論と今後の課題

五十嵐と小林の計算資源使用法解析の枠組み[6)] で未解決であった，各計算資源の使われ方を表す使用法表現が表す言語と資源の仕様を表す言語との間の包含関係を判定するためのアルゴリズムを提案した．本アルゴリズムにより，多くの計算資源に対して使用法の検証が自動化される．

今後の課題として，検証可能な仕様のクラスを広げることがあげられる．また，例外処理機構を持つプログラミング言語を扱うためには，使用法表現に例外に相当する構成子を追加して拡張する必要がある[8)]．そのような拡張を施した使用法表現についても，本論文と同様の手法で包含判定のアルゴリズムを構築することができると思われるが，その定式化や証明は今後の課題である．

謝辞 文脈自由文法どうしのシャッフルと正則言語との間の包含関係の判定が決定不能であることを示唆してくださった大山口通夫氏に感謝いたします．

参 考 文 献

1) Aiken, A., Fähndrich, M. and Levien, R.: Improving Region-Based Analysis of Higher-Order Languages, *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp.174–185 (1995).
2) Bigliardi, G. and Laneve, C.: A Type System for JVM Threads, *Proc. 3rd ACM SIGPLAN Workshop on Types in Compilation* (*TIC2000*), Montreal, Canada (2000).
3) Field, J., Goyal, D., Ramalingam, G. and Yahav, E.: Typestate verification: Abstraction techniques and complexity results, *Sci. Comput. Program*, Vol.58, No.1-2, pp.57–82 (2005).
4) Freund, S.N. and Mitchell, J.C.: The type system for object initialization in the Java bytecode language, *ACM Trans. Prog. Lang. Syst.*, Vol.21, No.6, pp.1196–1250 (1999).
5) Hopcroft, J.E., Motwani, R. and Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*, 2nd Edition, Addison Wesley (2000).
6) Igarashi, A. and Kobayashi, N.: Resource Usage Analysis, *ACM Trans. Prog. Lang. Syst.*, Vol.27, No.2, pp.264–313 (2005).
7) 岩間 太：型に基づく計算資源使用法の検証，博士論文，東北大学情報科学研究科（準備中）．
8) Iwama, F., Igarashi, A. and Kobayashi, N.: Resource Usage Analysis for Functional Language with Exceptions, *Proc. Partial Evaluation and Program Manipulation 06* (*PEPM'06*), ACM, pp.38–47, ACM Press (2006).
9) Iwama, F. and Kobayashi, N.: A New Type System for JVM Lock Primitives, *Proc. ASIA-PEPM'02*, ACM, pp.156–168, ACM Press (2002).
10) Kobayashi, N., Suenaga, K. and Wischik, L.: Resource Usage Analysis for the Pi-Calculus, *Logical Methods in Computer Science*, Vol.2, No.3:4, pp.1–42 (2006).
11) Minamide, Y.: Static Approximation of Dynamically Genarated Web Pages, *Proc. 14th International World Wide Web Conference*, LNCS, Vol.4279, ACM, pp.357–373, Springer-Verlag (2006).
12) Minamide, Y. and Tozawa, A.: XML Validation for Context-Free Grammars, *Proc. APLAS06*, LNCS 4279, pp.357–373 (2006).
13) Strom, R.E. and Yemini, S.: Typestate: A Programming Language Concept for Enhancing Software Reliability, *IEEE Trans. Softw. Eng.*,

Vol.12, No.1, pp.157–171 (1986).
14) Tofte, M. and Talpin, J.-P.: Region-based Memory Management, *Information and Computation*, Vol.132, No.2, pp.109–176 (1997).
15) Walker, D., Crary, K. and Morrisett, J.G.: Typed memory management via static capabilities, *ACM Trans. Prog. Lang. Syst.*, Vol.22, No.4, pp.701–771 (2000).
16) Xu, Z., Miller, B.P. and Reps, T.: Safety checking of machine code, *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp.70–82, ACM Press (2000).
17) Xu, Z., Reps, T.W. and Miller, B.P.: Typestate Checking of Machine Code, *ESOP '01: Proc. 10th European Symposium on Programming*, London, UK, pp.335–351, Springer-Verlag (2001).
18) 大山口通夫：文脈自由文法にシャッフル演算子を加えた文法の言語と正則言語との間の言語包含関係の決定不能性 (2005). Personal communication.

付　　録

A.1 補題 3.11 の証明

補題 A.1.1

(1)　$s \sim_M s' \ \wedge \ t \sim_M t' \Rightarrow st \sim_M s't'$

(2)　$s_1 \sim_M s_1' \wedge s_2 \sim_M s_2' \Rightarrow [\{s_1\} \otimes \{s_2\}]_{\sim_M} \cong_M [\{s_1'\} \otimes \{s_2'\}]_{\sim_M}$

証明 (1)：$\sim_M$ の定義より明らか.

(2)：$s \in [\{s_1\} \otimes \{s_2\}]_\sim$ とおく．$s \in [\{s_1'\} \otimes \{s_2'\}]_\sim$ または，$\exists s_e \in [\{s_1\} \otimes \{s_2\}]_\sim, \exists s_e' \in [\{s_1'\} \otimes \{s_2'\}]_\sim .(s_e \in Err(M) \wedge s_e' \in Err(M))$ を示せばよい.

このとき，s_{12} が存在して，$s_{12} \in \{s_1\} \otimes \{s_2\}$ かつ $s \sim s_{12}$ を満たす．ここで，s_1，s_2 が属している同値類の種類によって場合分けを行う．以降，3.3 節の同値類表現を使用する.

- $s_1 \in [\![\epsilon]\!]_M$ の場合：このとき $s_1 = \epsilon$．よって，$s_{12} = s_2$．$s_1 \sim s_1'$ より，$s_1' = \epsilon$．よって，$[\{s_1'\} \otimes \{s_2'\}]_\sim = [\{s_2'\}]_\sim = [\{s_2\}]_\sim = [\{s_{12}\}]_\sim = [\{s\}]_\sim \ni s$.
- $s_2 \in [\![\epsilon]\!]_M$ の場合：上の場合と同様.
- $s_1, s_2 \in [\![q]\!]_M$ の場合：$\forall s_0 \in \{s_1\} \otimes \{s_2\}.s_0 \in [\![q]\!]_M$．$\forall s_0' \in \{s_1\} \otimes \{s_2\}.s_0' \in [\![q]\!]_M$．よって，$s_{12} \in [\![q]\!]_M$ である．これより，$[\{s_1'\} \otimes \{s_2'\}]_\sim \supseteq [\![q]\!]_M \ni s_{12}$．$s_{12} \sim s$ より，$[\{s_1'\} \otimes \{s_2'\}]_\sim \ni s$.
- $s_1 \in [\![q_1]\!]_M$，$s_2 \in [\![q_2]\!]_M$，$q_1 \neq q_2$ の場合：$Q^M_{loop}(s_1) = \{q_1\}$，$Q^M_{loop}(s_2) = \{q_2\}$ より，$\forall s_0 \in \{s_1\} \otimes \{s_2\}.(s_0 \in [\![\mathbf{err}]\!]_M)$．$\forall s_0' \in \{s_1'\} \otimes \{s_2'\}.(s_0' \in [\![\mathbf{err}]\!]_M)$．ここで一般に，$s'' \in [\![\mathbf{err}]\!]_M$ ならば $s'' \in Err(M)$．よって，成り立つ.
- $s_1 \in [\![q_1]\!]_M$，$s_2 \in [\![q_2q_3]\!]_M$ の場合：$Q^M_{loop}(s_1) = \{q_1\}$，$Q^M_{path}(s_2) = \{(q_2, q_3)\}$，$Q^M_{loop}(s_2) \neq \{q2\}$ かつ $Q^M_{loop}(s_1') = \{q_1\}$，$Q^M_{path}(s_2') = \{(q_2, q_3)\}$，$Q^M_{loop}(s_2') \neq \{q2\}$ が成り立つ．よって，ある q_4，q_4'，s_{21}，s_{21}'，s_{22}，s_{22}' が存在して $q_4 \neq q_1$，$q_4' \neq q_1'$，$s_2 = s_{21}s_{22}$，$s_2' = s_{21}'s_{22}'$，$\hat{\delta}(q_1, s_{21}) = q_4$，$\hat{\delta}(q_4, s_{22}) = q_2$，$\hat{\delta}(q_1, s_{21}') = q_4'$，$\hat{\delta}(q_4, s_{22}') = q_2$ を満たす.
  このとき，$q_4 \neq q_1$ より，$s_1s_2 \in Err(M)$ もしくは $s_{21}s_1s_{22} \in Err(M)$ が成り立つ．同様に $q_4' \neq q_1'$ より，$s_1's_2' \in Err(M)$ もしくは $s_{21}'s_1's_{22}' \in Err(M)$ が成り立つ.
  $s_1s_2, s_{21}s_1s_{22} \in \{s_1\} \otimes \{s_2\}$，$s_1's_2', s_{21}'s_1's_{22}' \in \{s_1'\} \otimes \{s_2'\}$ この場合も成り立つ.
- $s_1 \in [\![q_1q_2]\!]_M$，$s_2 \in [\![q_3q_4]\!]_M$ の場合：上の場合と同様.
- $s_1 \in [\![\mathbf{err}]\!]_M$ の場合，$s_1s_2 \in \{s_1\} \otimes \{s_2\} \subseteq [\{s_1\} \otimes \{s_2\}]_\sim$ かつ $s_1's_2' \in \{s_1'\} \otimes \{s_2'\} \subseteq [\{s_1'\} \otimes \{s_2'\}]_\sim$ である．$s_1s_2, s_1's_2' \in Err(M)$ より，この場合も成り立つ.
- $s_2 \in [\![\mathbf{err}]\!]_M$ の場合，上記の場合と同様.　□

証明（補題 3.11）

(1) の証明：

$[\{t_1\}; \{t_2\}]_\sim \subseteq [[t_1]_\sim; [t_2]_\sim]_\sim$ は明らかなので，$t \in [[t_1]_\sim; [t_2]_\sim]_\sim$ に対して，$t \in [\{t_1\}; \{t_2\}]_\sim$ を示せば十分である.

$s \in [[t_1]_\sim; [t_2]_\sim]_\sim$ とすると $\exists s_{12}.(s_{12} \sim s \ \wedge \ s_{12} \in [t_1]_\sim; [t_2]_\sim)$.

; の定義より，s_{12} について次のどちらかが成り立つ.

- (a): $s_{12} = s_1s_2$，$s_1{\downarrow} \in [t_1]_\sim$，$s_2 \in [t_2]_\sim$
- (b): $s_{12} = s_1$，$s_1 \in [t_1]_\sim$

(a) の場合：$s_1{\downarrow} \in [t_1]_\sim$ より，$\exists s_1'.(t_1 = s_1'{\downarrow} \wedge s_1 \sim s_1')$，$s_2 \in [t_2]_\sim$ より，$\exists s_2'.(t_2 = s_2' \ \wedge \ s_2 \sim s_2')$．よって，$\{t_1\}; \{t_2\} = \{s_1's_2'\}$．ここで補題 A.1.1 (1) により，$[\{t_1t_2\}]_\sim = [\{s_1's_2'\}]_\sim = [\{s_1s_2\}]_\sim = [\{s_{12}\}]_\sim = [\{s\}]_\sim \ni s$．(b) の場合：$\exists s_1'.(t_1 = s_1' \ \wedge \ s_1 \sim s_1')$ が成り立つので，補題 A.1.1 (1) により，$[\{t_1\}; \{t_2\}]_\sim = [\{s_1'\}]_\sim = [\{s_1\}]_\sim = [\{s_{12}\}]_\sim = [\{s\}]_\sim \ni s$.

$s{\downarrow} \in [t_1]_\sim; [t_2]_\sim$ とすると $\exists s'.(s' \sim s \ \wedge \ s' \in [t_1]_\sim; [t_2]_\sim)$．; の定義より，次が成り立つ.

- $s_{12} = s_1s_2$，$s_1{\downarrow} \in [t_1]_\sim$，$s_2{\downarrow} \in [t_2]_\sim$.

よって，$\exists s_1'.(t_1 = s_1'{\downarrow} \ \wedge \ s_1 \sim s_1')$，$\exists s_2'.(t_2 = s_2'{\downarrow} \ \wedge \ s_2 \sim s_2')$．このとき，$[\{t_1\}; \{t_2\}]_\sim = [\{s_1's_2'{\downarrow}\}]_\sim = [\{s_1s_2{\downarrow}\}]_\sim = [\{s_{12}{\downarrow}\}]_\sim = [\{s{\downarrow}\}]_\sim \ni$

$s\downarrow$.

(2) の証明：

$[\{t_1\}\otimes\{t_2\}]_\sim \subseteq [[t_1]_\sim\otimes[t_2]_\sim]_\sim$ は明らかなので，

$t\in[[t_1]_\sim\otimes[t_2]_\sim]_\sim$ に対して，$t\notin[\{t_1\}\otimes\{t_2\}]_\sim$ ならば $\exists t'\in[[t_1]_\sim\otimes[t_2]_\sim]_\sim, \exists t''\in[\{t_1\}\otimes\{t_2\}]_\sim.(t'\in Err(M)\wedge t''\in Err(M))$ を示せばよい．

$s\in[[t_1]_\sim\otimes[t_2]_\sim]_\sim$ とする．このとき，$\exists s_{12}\in[t_1]_\sim\otimes[t_2]_\sim.(s_{12}\sim s)$．よって，ある s_1，s_2 が存在して，$s_1\sim t_1$ かつ $s_2\sim t_2$ かつ $s_{12}\in\{s_1\}\otimes\{s_2\}\subseteq[t_1]_\sim\otimes[t_2]_\sim$ を満たす．

よって，補題 A.1.1 (2) より，$[\{t_1\}\otimes\{t_2\}]_\sim \cong [\{s_1\}\otimes\{s_2\}]_\sim\subseteq[[t_1]_\sim\otimes[t_2]_\sim]_\sim$ が成り立つ．ここで，$s\notin[\{t_1\}\otimes\{t_2\}]_\sim$ ならば，$[\{t_1\}\otimes\{t_2\}]_\sim\cong[\{s_1\}\otimes\{s_2\}]_\sim$ より，$t'_{12}\in[\{t_1\}\otimes\{t_2\}]_\sim$，$t''_{12}\in[\{s_1\}\otimes\{s_2\}]_\sim\subseteq[[t_1]_\sim\otimes[t_2]_\sim]_\sim$ が存在して，t'_{12}，$t''_{12}\in Err(M)$ を満たす．よって証明された．

$s\downarrow\in[[t_1]_\sim\otimes[t_2]_\sim]_\sim$ の場合も同様に示される．□

A.2 補題 3.15 の証明

証明（補題 3.15）

$\forall\rho\in dom(F)=dom(F').[\![F(\rho)]\!]_{\sim_M}\cong_M\alpha_M(F'(\rho))$ を満たす任意の F，F' について以下が成り立つことを示せば十分である．

$[\![U]\!]_F]_{\sim_M}\cong_M\alpha_M^{F'}(U)$

U の構造に関する帰納法．

- $U=\mathbf{0},l$ の場合：
  $[\![\cdot]\!]_F$，$\alpha_M^{F'}(\cdot)$ の定義より成り立つ．
- $U=\rho$ の場合：F，F' の条件より成り立つ．
- $U=U_1;U_2$ の場合：
  $\alpha_M^{F'}(U_1;U_2)=[\alpha_M^{F'}(U_1);\alpha_M^{F'}(U_2)]_\sim$ である．帰納法の仮定より，$[\alpha_M^{F'}(U_1);\alpha_M^{F'}(U_2)]_\sim\cong_M[[[\![U_1]\!]_F]_\sim;[[\![U_2]\!]_F]_\sim]_\sim$ が，上記条件を満たす F について成り立つ．補題 3.11 より，
  $$[[[\![U_1]\!]_F]_\sim;[[\![U_2]\!]_F]_\sim]_\sim\cong_M[[\![U_1]\!]_F;[\![U_2]\!]_F]_\sim\cong_M[[\![U_1;U_2]\!]_F]_\sim$$
  よって成り立つ．
- $U=U_1\otimes U_2$，$U_1\&U_2$ の場合：
  $U=U_1;U_2$ の場合と同様．
- $U=\mu\rho.U$ の場合：
  $$[\![U]\!]_F=\mathbf{lfp}(\lambda x.[\![U]\!]_{F\{\rho\mapsto x\}})$$
  $$\alpha_M^{F'}(U)=\mathbf{lfp}(\lambda x.\alpha_M^{F'\{\rho\mapsto x\}}(U)$$
  である．ここで，$G(x)=[\![U]\!]_{F\{\rho\mapsto x\}}$，$H(x)=\alpha_M^{F'\{\rho\mapsto x\}}(U)$，
  $$A^{(0)}=G(\{\epsilon\})\qquad B^{(0)}=H(\{\epsilon\})$$
  $$A^{(i+1)}=G(A^{(i)})\qquad B^{(i+1)}=H(B^{(i)})$$
  とおくと，各 i について $[A^{(i)}]_\sim\cong_M B^{(i)}$ がいえればよい．i に関する数学的帰納法で示す．
  - $i=0$：$[\![\emptyset]\!]_\sim=\alpha_M(\emptyset)=\emptyset$ と構造的帰納法の仮定より成り立つ．
  - $[A^{(k)}]_\sim\cong_M B^{(k)}$ を仮定したときの $i=k+1$ の場合：
    $$\forall\rho\in dom(F\{\rho\mapsto A^{(k)}\})=dom(F'\{\rho\mapsto B^{(k)}\}).([\![F(\rho)]\!]_{\sim_M}\cong_M\alpha_M(F'(\rho)))$$
    が $[A^{(k)}]_\sim\cong_M B^{(k)}$ と数学的帰納法の仮定より成り立ち，その結果，構造的帰納法の仮定より，$[A^{(k+1)}]_\sim\cong_M B^{(k+1)}$ が成り立つ．□

A.3 $[\![U]\!]\subseteq(R\downarrow)^\#$ の決定不能性

本節では，仕様のクラスを一般の正規言語のクラスに広げた場合，使用法表現が表す言語との包含関係は決定不能であること（定理 A.3.1）を示す．

定理 A.3.1 U を使用法表現，R を正則言語とすると，

$[\![U]\!]\subseteq(R\downarrow)^\#$

は決定不能である．

上記の結果は，以下の決定不能性の結果から導かれる．なお，$L_1\star L_2$ は言語 L_1 と L_2 のシャッフルを表すものとする．

補題 A.3.2（大山口[18]） L_1，L_2 を文脈自由言語，R を正則言語とすると

$L_1\star L_2\subseteq R$

は決定不能．

証明 付録 A.4 を参照. □

定理 A.3.1 は，補題 A.3.2 の決定不能問題を定理 A.3.1 の判定問題に還元することにより証明できる．まず補題をいくつか準備する．

補題 A.3.3 $L_1,L_2\subseteq\mathcal{L}^*$ かつ $\downarrow\notin\mathcal{L}$ とする．$L_1\subseteq L_2$ と $(L_1\downarrow)^\#\subseteq(L_2\downarrow)^\#$ は同値である．

証明

- $L_1\subseteq L_2\Rightarrow(L_1\downarrow)^\#\subseteq(L_2\downarrow)^\#$:
  $L_1\subseteq L_2$ かつ $t\in(L_1\downarrow)^\#$ を仮定する．$t\in(L_1\downarrow)^\#$ より，$t=s\downarrow$ または $t\in\mathcal{L}^*$ が成り立つ．前者の場合，$s\in L_1\subseteq L_2$．よって，$t\in L_2\downarrow\subseteq(L_2\downarrow)^\#$．後者の場合，ある s_1 について $ts_1\downarrow\in L_1\downarrow\subseteq L_2\downarrow$ が成り立つ．よって，$t\in(L_2\downarrow)^\#$．
- $(L_1\downarrow)^\#\subseteq(L_2\downarrow)^\#\Rightarrow L_1\subseteq L_2$:
  $(L_1\downarrow)^\#\subseteq(L_2\downarrow)^\#$ かつ $s\in L_1$ を仮定する．すると，$s\downarrow\in(L_1\downarrow)^\#\subseteq(L_2\downarrow)^\#$ が成り立つ．L_2 は $\downarrow$ を含まないので，$s\downarrow\in L_2\downarrow$，よって $s\in L_2$

が成り立つ．□

補題 A.3.4 $L \subseteq \mathcal{L}^*$ かつ $\downarrow \notin \mathcal{L}$ とする．L が空でない文脈自由言語ならば，$[\![U]\!] = (L\downarrow)^{\#}$ を満たす使用法表現 U が存在する．

証明（概略）L を生成する文脈自由文法を $G = (S, \mathcal{L}, N, V)$ とする．G の非終端記号はすべて生成的である（すなわち，その記号から生成される語が存在する）と仮定してよい[5]．G の各生成規則（すなわち V の要素）

$X \to \alpha$

を使用法表現の再帰方程式

$\rho_X = h(\alpha)$

で置き換えることができる．ここで，ρ_X は非終端記号 X に対応する usage 変数であり，$h(\alpha)$ は，$h(\epsilon) = \mathbf{0}$，$h(Y\alpha) = \rho_Y; h(\alpha)$，$h(b\alpha) = b; h(\alpha)$ により定義される．得られた連立再帰方程式の解[6] の，ρ_S の値が求める使用法表現である．□

以上の結果を用いて定理 A.3.1 を証明する．

証明（定理 A.3.1） L_1，L_2 を文脈自由文法，R を正規言語とする．補題 A.3.2 より，$L_1 \star L_2 \subseteq R$ が使用法表現と正規言語との包含関係に帰着できることを示せばよい．まず，補題 A.3.3 により，$L_1 \star L_2 \subseteq R$ は，

$((L_1 \star L_2)\downarrow)^{\#} \subseteq (R\downarrow)^{\#}$

と等価である．これはさらに，$\star$ と $\otimes$ の定義（定義 2.4）より，

$(L_1\downarrow)^{\#} \otimes (L_2\downarrow)^{\#} \subseteq (R\downarrow)^{\#}$

と等価である．ここで，補題 A.3.4 より，$(L_1\downarrow)^{\#} = [\![U_1]\!]$ かつ $(L_2\downarrow)^{\#} = [\![U_2]\!]$ を満たす U_1，U_2 が存在する．$(L_1\downarrow)^{\#} \otimes (L_2\downarrow)^{\#} = [\![U_1 \otimes U_2]\!]$ より，$L_1 \star L_2 \subseteq R$ は，$[\![U_1 \otimes U_2]\!] \subseteq (R\downarrow)^{\#}$ と等価．したがって，$L_1 \star L_2 \subseteq R$ の決定不能性（補題 A.3.2）より，$[\![U]\!] \subseteq (R\downarrow)^{\#}$ も決定不能である．□

A.4 文脈自由文法にシャッフル演算子加えた文法の言語と正則言語の包含判定問題の決定不能性

大山口氏の許可を得て，補題 A.3.2 の証明の概略を掲載する．

証明（補題 A.3.2） この問題は**ポストの対応問題**の決定不能性に帰着することができる．

ポストの対応問題とは，

$P = \{(u_1, v_1), \ldots, (u_n, v_n)\}, \quad u_i, v_i \in \{a, b\}^*$

が与えられたときに

$u_{i_1} \ldots u_{i_k} = v_{i_l} \ldots v_{i_k}$

となる空でない $i_1, \ldots, i_k$ が存在するかを判定する問題で，決定不能である．

L_1 を

$S \to \epsilon \mid 1\$SU_1 \mid \ldots \mid n\$SU_n$

$U_i \to a_{ik_i}\$ \ldots a_{i1}\$$ （if $u_i = a_{i1} \ldots a_{ik_i}$）

L_2 を

$S \to \epsilon \mid 1\$SV_1 \mid \ldots \mid n\$SV_n$

$V_i \to b_{ij_i}\$ \ldots b_{i1}\$$ （if $v_i = b_{i1} \ldots a_{ij_i}$）

によって生成される言語とする．ただし，$1, \ldots, n, \$$ は u_i，v_i（$i = 1, \ldots, n$）に出現しないものとする．

また R' を $\{xx\$\$ \mid x \in \{1, \ldots, n, a, b\}\}^+$ とし，R をその補集合とする．

すると，

$(L_1 \star L_2) \subseteq R$

$\Leftrightarrow (L_1 \star L_2) \cap R' = \emptyset$

$\Leftrightarrow$ ポストの対応問題 P が解を持たない

が成立する（ここで，$(L_1 \star L_2) \cap R'$ の要素は，$i_1 i_1 \$\$ \cdots i_k i_k \$\$ s$，$s \in (aa\$\$ + bb\$\$)^*$ 形をしており，$i_1, \ldots, i_k$ がポストの対応問題の解であることに注意されたい）．したがって，ポストの対応問題の決定不能性より，$L_1 \star L_2 \subseteq R$ も決定不能性である．

(平成 18 年 9 月 12 日受付)
(平成 18 年 12 月 10 日採録)

岩間　太（学生会員）

2001 年東京大学理学部情報科学科卒業．2004 年東京工業大学大学院情報理工学研究科修士課程修了．2006 年現在，東北大学大学院情報科学研究科博士課程在籍．日本ソフトウェア科学会会員．プログラム解析・検証，型システム等に興味を持つ．

五十嵐　淳

1995 年東京大学理学部情報科学科卒業．1997 年同大学大学院理学系研究科情報科学専攻修士課程修了．2000 年同大学院理学系研究科情報科学専攻博士課程修了．博士（理学）．同年東京大学大学院総合文化研究科助手．2002 年京都大学大学院情報学研究科講師．2006 年同助教授．ACM，日本ソフトウェア科学会各会員．プログラミング言語の基礎理論，特に型システムに興味を持つ．2006 年日本 IBM 科学賞受賞．

小林 直樹（正会員）

1968 年生．1991 年東京大学理学部情報科学科卒業．1993 年同大学大学院理学系研究科情報科学専攻修士課程修了，同年博士課程進学．東京大学大学院理学系研究科情報科学専攻助手，講師，東京工業大学大学院情報理工学研究科助教授を経て 2004 年より東北大学大学院情報科学研究科教授，現在に至る．博士（理学）．型理論，プログラム解析，並行計算等に興味を持つ．ACM 会員．2001 年 IFIP TC2 Manfred Paul Award，2003 年日本 IBM 科学賞受賞．