関係文字列を用いた協調実行言語の意味論

本多, 和正
九州大学大学院システム情報科学府情報理学専攻 : 博士後期課程

河原, 康雄
九州大学大学院システム情報科学研究院情報理学部門

https://doi.org/10.15017/1515715

出版情報：九州大学大学院システム情報科学紀要. 6 (1), pp.47-51, 2001-03-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第6巻　第1号　平成13年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.6, No.1, March 2001

関係文字列を用いた協調実行言語の意味論

本多和正* ・ 河原康雄**

Semantics of a Simple Coordination Language by Relational Strings

Kazumasa HONDA and Yasuo KAWAHARA

(Received December 1, 2000)

Abstract: The purpose of this study is to apply the denotational semantics of the coordination language to the safety analysis of parallel programs. In this paper, the denotational semantics of a simple coordination language is defined by relational strings, which are strings of binary relations, and the concatenation of relational strings is a timed-composition of binary relations.

Keywords: Coordination language, Semantics, Process algebra, Relational algebra

1. はじめに

本論文では関係文字列(Relational strings)を用いて並列プログラム,特に協調実行言語(Coordination language)の意味論を構成する.関係文字列とは,一般の文字列において文字を二項関係に置き換えたものである.しかし,関係文字列は単に置き換えただけではなく関係の性質を保存するように構成される.具体的には,文字列の連結演算を時系列構造を持った関係の合成演算と考えるのである.逐次プログラムの場合はプログラムの実行順序は1つしかないので,わざわざ時系列構造を考える必要はない.しかし並列プログラムの場合はそれぞれのプロセス内では1つだが,他のプロセス内も合わせてみてやると実行順序は複数ある.例えば共有変数 x,y が0で初期化された後 以下の2つのプロセスが並列に動くことを考える.

- Proc.A: $x = y + 1$.
- Proc.B: $y = 2$.

Aが先に実行された場合は,yの値は0のままであるので $x = 0 + 1 = 1$となる.しかし,Bが先に実行されると,yの値が2になるので $x = 2 + 1 = 3$となり 実行結果が異なる.AとBがまったく同時に実行される場合もあるがその場合でも yの値は0か2である.逐次型・並列型に限らず実行順序は実行結果に大きく影響するが,特に並列プログラムの場合,実行順序は複数ありそのうちのどれが実際の実行順序となるかは一般に非決定的である.このためそれらの実行順序を自由に再構成できる時系列構造が必要になる.そして再構成する場合の最小構成単位は,現在の状態を次の状態へ遷移させる関数である.本論文における表示的意味論は,プログラムに対し遷移関数の列の集合を対応付けることで構成する.

平成12年12月1日受付

* 情報理学専攻博士後期課程
** 情報理学部門

2. 関係文字列と演算

ここでは関係文字列とそれに対する演算の定義を行う.まずその準備として 文字列の定義と文字列に対する演算の定義を述べる.

2.1 文字列と演算

Σ を未定義シンボル！を持つ文字集合とする.Σ^* は Σ の文字列の集合である.以下 空文字列を ε と書き文字列 $\vec{\alpha}$ の成分表示を $\alpha_1.\alpha_2.\cdots.\alpha_n$ と書くことにする.

定義 1　$\forall\vec{\alpha},\vec{\beta}\in\Sigma^*$ に対し以下の操作を定義する.

- $\vec{\alpha}.\varepsilon = \varepsilon.\vec{\alpha} = \vec{\alpha}$.
- $\vec{\alpha}.! = !.\vec{\alpha} = !$.
- $(\bigcup_i\{\vec{\alpha_i}\}).(\bigcup_j\{\vec{\beta_j}\}) := \bigcup_{i,j}\{\vec{\alpha_i}.\vec{\beta_j}\}$.
- *left derivation,*

$$\vec{p}\backslash\vec{\alpha} := \begin{cases} \vec{\alpha'} & s.t.\ \ \vec{\alpha} = \vec{p}.\vec{\alpha'}, \\ ! & o.w. \end{cases}$$

$(\bigcup_j\{\vec{p_j}\})\backslash(\bigcup_i\{\vec{\alpha_i}\}) := \bigcup_{i,j}\{\vec{p_j}\backslash\vec{\alpha_i}\}$.

- *interleave composition,*

$|:\Sigma^*\times\Sigma^*\to\wp(\Sigma^*)$,

  - $\vec{\alpha}|\varepsilon := \varepsilon|\vec{\alpha} = \{\vec{\alpha}\}$.
  - $(a.\vec{\alpha})|(b.\vec{\beta}) := \{a\}.(\vec{\alpha}|b.\vec{\beta})\cup\{b\}.(a.\vec{\alpha}|\vec{\beta})$.
    ここで $a,b\in\Sigma$.

$(\bigcup_i\{\vec{\alpha_i}\})|(\bigcup_j\{\vec{\beta_j}\}) := \bigcup_{i,j}(\vec{\alpha_i}|\vec{\beta_j})$.

この定義から以下の性質が言える.

事実 2　$a\in\Sigma$, $\vec{\alpha},\vec{\beta},\vec{p},\vec{q}\in\Sigma^*$とすると以下が成り立つ.

$\varepsilon\backslash\vec{\alpha} = \vec{\alpha}$, $\vec{\alpha}\backslash\varepsilon = !\backslash\vec{\alpha} = \vec{\alpha}\backslash! = !$,

$\vec{\alpha}\neq! \Rightarrow \vec{\alpha}\backslash\vec{\alpha} = \varepsilon$,

$(\vec{p}.\vec{q})\backslash\vec{\alpha} = \vec{q}\backslash(\vec{p}\backslash\vec{\alpha})$,

$\{\vec{p}\}\backslash\{\vec{\alpha}.\vec{\beta}\} = \{(\vec{p}\backslash\vec{\alpha}).\vec{\beta}\}\cup\{(\vec{\alpha}\backslash\vec{p})\backslash\vec{\beta}\}$,

$\{a\}\backslash(\vec{\alpha}|\vec{\beta}) = (a\backslash\vec{\alpha})|\vec{\beta}\cup\vec{\alpha}|(a\backslash\vec{\beta})$,

$\{a\}.(\vec{\alpha}|\vec{\beta_j}) \subseteq (a.\vec{\alpha})|\vec{\beta}$,

$\{a\}.(\bigcup_i\{\vec{\alpha_i}\})|(\bigcup_j\{\vec{\beta_j}\}) \subseteq (\{a\}.\bigcup_i\{\vec{\alpha_i}\})|(\bigcup_j\{\vec{\beta_j}\})$.

2.2 関係文字列

簡単にいうと 関係文字列とはΣとして集合 X 上の関係の集合を持ってきた文字列である. ここで！は空関係 $\emptyset$ である. しかし ただ単純にΣを集合 X 上の関係の集合に置き換えたわけではなく, 連結演算を時系列構造をもった関係の合成演算として取り扱う. 定義3は関係文字列から時系列構造を取り去る操作の定義である.

定義 3　$\vec{\alpha} = \alpha_0.\vec{\alpha_1}$ とするとき

$\lceil\varepsilon\rceil := id$, $\lceil\alpha_0\rceil := \alpha_0$,

$\lceil\vec{\alpha}\rceil = \lceil\alpha_0.\vec{\alpha_1}\rceil := \alpha_0\lceil\vec{\alpha_1}\rceil$,

$\lceil\cup_i\{\vec{\alpha_i}\}\rceil := \sqcup_i\lceil\vec{\alpha_i}\rceil$.

事実 4　$\lceil\vec{\alpha}.\vec{\beta}\rceil = \lceil\vec{\alpha}\rceil\lceil\vec{\beta}\rceil$.

3. Simple-CL

本論文では並列プログラム言語として単純な協調実行言語 Simple-CLを考える. 協調実行言語とはソフトウェアコンポーネントのデザインから相互作用[†1] の分離を支援する言語であり既存の逐次型プログラム言語にタプル空間というデータベース的な振舞いをする共有メモリ(に対する操作)を追加して作るのが一般的である.

本論文で用いるSimple-CLには, 動的なプロセス生成を行わない, 反復(while文)を含まない, パターンマッチングを行わない, という制限がある. しかし協調実行言語の本質的な特徴はもっている. 本論文では共有メモリ(タプル空間)の状態変化によってSimple-CLに意味を与える. 状態変化による直観的な意味付けとしては状態のトレースを用いたものが挙げられる. しかし 単に状態のトレースを集めただけでは良くないことがわかっている[1]. これはプログラムの命令は状態を遷移させる関数であるのに状態のトレースではその関数の一部しか現れないために起こる. 例えば 本来は与えられた整数に1を加える命令であったのに実際に実行されたのは4に1を加える操作であったため$\{(x,x+1)|\forall x \in \mathbf{Z}\}$であるはずのものがトレースでは$\{(4,5)\}$になってしまう. これでは他のプロセスとうまく合成できないのは当然である.

タプル空間操作(通信)の集合を$\mathcal{C}$, タプルの集合を$\mathcal{T}$, タプル空間の状態の集合を$\mathcal{S}$とし[†2], $\mathcal{S}$は存在しないタプルを取ろうとした(過剰削除)状態を表す$\perp$を持つとする. また, タプル空間が有限の場合には過剰追加状態$\top$も持つとしタプル空間操作を表す$\mathcal{S}$から$\mathcal{S}$への全域関数の部分集合$\mathcal{F}$を以下のように定義する.

(タプル空間が無限の場合)

$\mathcal{F} := \{f[c] \mid c \in \mathcal{C}, f[c](\perp) = \perp\}$.

(タプル空間が有限の場合)

$\mathcal{F} := \{f[c] \mid c \in \mathcal{C}, f[c](\perp) = \perp, f[c](\top) = \top\}$.

例 5　基本的なタプル空間操作の例.(タプル空間は無限)

$\mathcal{C} := \{put(t), get(t), probe(t) \mid t \in \mathcal{T}\}$,

$\mathbf{t} := \{(\sigma, \sigma \cup \{t\}) \in \mathcal{S} \times \mathcal{S} \mid \sigma \neq \perp\}$,

(追加) $f[put(t)] := \mathbf{t} \sqcup \{(\perp, \perp)\}$,

(取得) $f[get(t)] := \mathbf{t}^\sharp \sqcup \{(\sigma, \perp) \mid t \notin \sigma \vee \sigma = \perp\}$,

(探査) $f[probe(t)] := \mathbf{t}^\sharp\mathbf{t} \sqcup \{(\sigma, \perp) \mid t \notin \sigma \vee \sigma = \perp\}$.

定義 6 (Simple-CL)　*Simple-CL*の文の集合$\mathcal{A}$を以下の*BNF*で与える.

$$\mathcal{A} ::= \mathcal{C} \mid E \mid \mathcal{A};\mathcal{A} \mid \mathcal{A} + \mathcal{A} \mid \mathcal{A}\|\mathcal{A}.$$

以降特に断りのない場合$c \in \mathcal{C}$, $A, B, C, A', B' \in \mathcal{A}$ $\sigma \in \mathcal{S}$ であるとする.

定義 7 (遷移規則)　$\varepsilon(\sigma) = \sigma$とする. このとき, 遷移関係$\longrightarrow$を次のように定義する.

$$\langle c,\sigma\rangle \xrightarrow{f[c]} \langle E, f[c](\sigma)\rangle, \qquad \langle E;A,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle,$$
$$\langle E\|A,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle, \qquad \langle A\|E,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle,$$
$$\langle A+B,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle, \qquad \langle A+B,\sigma\rangle \xrightarrow{\varepsilon} \langle B,\sigma\rangle,$$
$$\langle A,\sigma\rangle \xrightarrow{x} \langle A', x(\sigma)\rangle \Rightarrow \langle A;B,\sigma\rangle \xrightarrow{x} \langle A';B, x(\sigma)\rangle,$$
$$\langle A,\sigma\rangle \xrightarrow{x} \langle A', x(\sigma)\rangle \Rightarrow \langle A\|B,\sigma\rangle \xrightarrow{x} \langle A'\|B, x(\sigma)\rangle,$$
$$\langle A,\sigma\rangle \xrightarrow{x} \langle A', x(\sigma)\rangle \Rightarrow \langle B\|A,\sigma\rangle \xrightarrow{x} \langle B\|A', x(\sigma)\rangle.$$

選択$A + B$は"if cond then A else B"の解釈であるので厳密に言うとcondの評価が必要である. しかし condの評価値はそれぞれのプロセスの内部状態にも依存しそのプロセスが教えない限り 他のプロセスは知ることが出来ない. もちろん タプル空間もcondの評価値を知らない. そこで今回はcondの評価を省略し 考えうる全ての遷移を扱う.

上記の遷移は1ステップのものであるが 動作を考える場合複数回の遷移をまとめて見たほうが良い時もある.

そこで以下のように複数回の遷移を定義する.

定義 8　複数回遷移関係$\longrightarrow\!\!\!\!\!\rightarrow$を以下のように定義する.

$$\langle A,\sigma\rangle \xrightarrow{x} \langle B, x(\sigma)\rangle \Rightarrow \langle A,\sigma\rangle \xrightarrow{x}\!\!\!\!\!\rightarrow \langle B, x(\sigma)\rangle.$$
$$\begin{cases} \langle A,\sigma\rangle \xrightarrow{\vec{x}}\!\!\!\!\!\rightarrow \langle B, \lceil\vec{x}\rceil(\sigma)\rangle, \\ \langle B, \lceil\vec{x}\rceil(\sigma)\rangle \xrightarrow{y} \langle C, y(\lceil\vec{x}\rceil(\sigma))\rangle, \end{cases} \Rightarrow \langle A,\sigma\rangle \xrightarrow{\vec{x}.y}\!\!\!\!\!\rightarrow \langle B, \lceil\vec{x}.y\rceil(\sigma)\rangle.$$

この複数回遷移は以下の性質を持つ.

事実 9　$\langle A,\sigma\rangle \xrightarrow{\vec{x}}\!\!\!\!\!\rightarrow \langle B, \lceil\vec{x}\rceil(\sigma)\rangle$のとき以下が成り立つ.

$$\langle A;C,\sigma\rangle \xrightarrow{\vec{x}}\!\!\!\!\!\rightarrow \langle B;C, \lceil\vec{x}\rceil(\sigma)\rangle,$$
$$\langle A+C,\sigma\rangle \xrightarrow{\vec{x}}\!\!\!\!\!\rightarrow \langle B, \lceil\vec{x}\rceil(\sigma)\rangle,$$
$$\langle A\|C,\sigma\rangle \xrightarrow{\vec{x}}\!\!\!\!\!\rightarrow \langle B\|C, \lceil\vec{x}\rceil(\sigma)\rangle.$$

4. 表示的意味論

先程定義した遷移規則に基づいてSimple-CLの表示的意味論を作る. なお この定義が遷移規則に正しく対応することについては4.1節で証明する.

---

†1 相互作用とはプロセス間の明示的な通信だけでなく共有メモリを介した暗黙的なデータ交換も含んでいる.

†2 一般には$\mathcal{S}$は重複を許した$\mathcal{T}$の部分集合である.

定義 10 $(\mathcal{D}:\mathcal{A}\to\wp(\mathcal{F}^*))$ *Simple-CL*の表示的意味$\mathcal{D}$を以下のように定義する.

$$\mathcal{D}(E):=\{\varepsilon\},\quad \mathcal{D}(A;B):=\mathcal{D}(A).\mathcal{D}(B),$$
$$\mathcal{D}(c):=\{f[c]\},\ \mathcal{D}(A+B):=\mathcal{D}(A)\cup\mathcal{D}(B),$$
$$\mathcal{D}(A\|B):=\mathcal{D}(A)|\mathcal{D}(B).$$

文字列集合に対する演算の定義から以下のような結合律,分配律などがすぐに言える.

事実 11 任意の$A,B,C\in\mathcal{A}$に対し以下が成り立つ.

$$\begin{aligned}
&\mathcal{D}((A;B);C) &&=\mathcal{D}(A;(B;C)),\\
&\mathcal{D}((A+B)+C) &&=\mathcal{D}(A+(B+C)),\\
&\mathcal{D}((A\|B)\|C) &&=\mathcal{D}(A\|(B\|C)),\\
&\mathcal{D}(A\|B) &&=\mathcal{D}(B\|A),\\
&\mathcal{D}(A+B) &&=\mathcal{D}(B+A),\\
&\mathcal{D}(A;(B+C)) &&=\mathcal{D}(A;B+A;C),\\
&\mathcal{D}((A+B);C) &&=\mathcal{D}(A;C+B;C),\\
&\mathcal{D}((A+B)\|C) &&=\mathcal{D}(A\|C+B\|C),\\
&\mathcal{D}(A;E) &&=\mathcal{D}(A),\\
&\mathcal{D}(E;A) &&=\mathcal{D}(A),\\
&\mathcal{D}(A\|E) &&=\mathcal{D}(A).
\end{aligned}$$

Aに関する構造帰納法により 以下のことが言える.

事実 12 $\mathcal{D}(A)\neq\emptyset$

定義 13 (遷移可能) $A\neq B$のとき, $\vec{p}.\mathcal{D}(B)\subseteq\mathcal{D}(A)$となる$\vec{p}\in\mathcal{F}^*$が存在するならば(表示的意味$\mathcal{D}$において) AはBに遷移可能であると呼ぶ. また$\vec{p}\in\mathcal{F}^+$に対し$\vec{p}\backslash\mathcal{D}(A)\neq\emptyset$のとき($\mathcal{D}$において) Aは$\vec{p}$で遷移可能であるという.

4.1 遷移規則と表示的意味の対応

定義10による表示的意味が遷移規則と正しく対応することを示す. 示すべきことは 遷移規則で得られるラベルの列と表示的意味に含まれる列が一致することである(定理17).

定理 14 $A\neq E$のとき以下が成り立つ.

$$\mathcal{D}(E)\subseteq\mathcal{D}(A)\Leftrightarrow\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle.$$

定理 15 $\mathcal{D}$において遷移可能であれば遷移規則で遷移する. つまり $p\in\mathcal{F}$ に対し

$$p\backslash\mathcal{D}(A)\neq\emptyset\Rightarrow\ \exists B\ s.t.\ \langle A,\sigma\rangle\xrightarrow{p}\!\!\!\!\rightarrow\langle B,p(\sigma)\rangle$$
更に $p.\mathcal{D}(B)\subseteq\mathcal{D}(A)$.

定理 16 遷移規則において遷移すれば$\mathcal{D}$において遷移可能である. つまり $p\in\mathcal{F}\cup\{\varepsilon\}$ に対し

$$\exists B\ s.t.\ \langle A,\sigma\rangle\xrightarrow{p}\langle B,p(\sigma)\rangle\Rightarrow p.\mathcal{D}(B)\subseteq\mathcal{D}(A).$$

なお 先の事実により$p\backslash\mathcal{D}(A)\supseteq\mathcal{D}(B)\neq\emptyset$である.

定理14～16を複数回用いることで定理17を得る.

定理 17 $A\neq E,\vec{p}\in\mathcal{F}^*$のとき以下が成り立つ.

$$\langle A,\sigma\rangle\xrightarrow{\vec{p}}\!\!\!\!\rightarrow\langle E,\lceil\vec{p}\rceil(\sigma)\rangle\Leftrightarrow\vec{p}\in\mathcal{D}(A).$$

以下は定理14～17の証明である. 構文と遷移規則に関する構造帰納法によって証明を行っている.

証明 (定理14)

$\mathcal{D}$の定義から $\mathcal{D}(E)\subseteq\mathcal{D}(A)\Leftrightarrow\varepsilon\in\mathcal{D}(A)$である.

- $\varepsilon\in\mathcal{D}(A)\Rightarrow\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$の証明.
  - $\varepsilon\in\mathcal{D}(A;B)$のとき.
    $\mathcal{D}(A;B)=\mathcal{D}(A).\mathcal{D}(B)$であるので連結演算と$\varepsilon$の定義により $\varepsilon\in\mathcal{D}(A)$ かつ $\varepsilon\in\mathcal{D}(B)$.
    ここで $A=E$の場合は $\langle E;B,\sigma\rangle\xrightarrow{\varepsilon}\langle B,\sigma\rangle$.
    $A\neq E$の場合は帰納法の仮定から$\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$. 遷移規則により$\langle A;B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle B,\sigma\rangle$.
    どちらの場合も $B=E$ならば自明.
    $B\neq E$のときは帰納法の仮定から$\langle A;B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$.
  - $\varepsilon\in\mathcal{D}(A+B)$のとき.
    $\mathcal{D}(A+B)=\mathcal{D}(A)\cup\mathcal{D}(B)$であるので
    $\varepsilon\in\mathcal{D}(A)$の場合を考えればよい.
    $A=E$の場合は $\langle E+B,\sigma\rangle\xrightarrow{\varepsilon}\langle E,\sigma\rangle$.
    $A\neq E$の場合は仮定から$\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$.
    遷移規則により $\langle A+B,\sigma\rangle\xrightarrow{\varepsilon}\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$
  - $\varepsilon\in\mathcal{D}(A\|B)$のとき.
    $\mathcal{D}(A\|B)=\mathcal{D}(A)|\mathcal{D}(B)$であるので"|"と$\varepsilon$の定義により $\varepsilon\in\mathcal{D}(A)$ かつ $\varepsilon\in\mathcal{D}(B)$.
    ここで $A=E$の場合は $\langle E\|B,\sigma\rangle\xrightarrow{\varepsilon}\langle B,\sigma\rangle$.
    $A\neq E$の場合は帰納法の仮定から$\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$. 遷移規則により$\langle A\|B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle B,\sigma\rangle$.
    どちらの場合も $B=E$ならば自明.
    $B\neq E$のときは帰納法の仮定から$\langle A\|B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$.
- $\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle\Rightarrow\varepsilon\in\mathcal{D}(A)$の証明.
  - $\langle A;B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$のとき.
    $A\neq E\wedge B\neq E$の場合は遷移規則から
    $\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle\wedge\langle B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$.
    帰納法の仮定により $\varepsilon\in\mathcal{D}(A)\wedge\varepsilon\in\mathcal{D}(B)$.
    よって $\varepsilon=\varepsilon.\varepsilon\in\mathcal{D}(A).\mathcal{D}(B)=\mathcal{D}(A;B)$.
    $A=E\wedge B\neq E$ の場合は $\mathcal{D}(A)=\{\varepsilon\}$ かつ $\langle B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$. 仮定により $\varepsilon\in\mathcal{D}(B)$.
    よって $\varepsilon=\varepsilon.\varepsilon\in\mathcal{D}(A).\mathcal{D}(B)=\mathcal{D}(A;B)$.
    $A=B=E$や$A\neq E\wedge B=E$の場合も同様.
  - $\langle A+B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$のとき.
    $A=E$または$\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$の場合を考えればよい. $A=E$のときは自明. 後者の場合は仮定から$\varepsilon\in\mathcal{D}(A)\subseteq\mathcal{D}(A+B)$.
  - $\langle A\|B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$のとき.
    $A\neq E\wedge B\neq E$の場合は$\langle A,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$ かつ $\langle B,\sigma\rangle\xrightarrow{\varepsilon}\!\!\!\!\rightarrow\langle E,\sigma\rangle$. 仮定から $\varepsilon\in\mathcal{D}(A)$ かつ $\varepsilon\in\mathcal{D}(B)$. よって $\varepsilon=\varepsilon.\varepsilon\in\mathcal{D}(A)|\mathcal{D}(B)=\mathcal{D}(A\|B)$.
    $A=E\wedge B\neq E$ の場合は $\mathcal{D}(A)=\{\varepsilon\}$ かつ

$\langle B,\sigma\rangle \xrightarrow{\varepsilon} \langle E,\sigma\rangle$. 仮定により $\varepsilon \in \mathcal{D}(B)$.
よって $\varepsilon = \varepsilon.\varepsilon \in \mathcal{D}(A)|\mathcal{D}(B) = \mathcal{D}(A\|B)$.
$A = B = E$や$A \neq E \wedge B = E$の場合も同様. ■

証明 (定理 15)

$\mathcal{C}$に関しては自明.

- $p\backslash\mathcal{D}(A;B) \neq \emptyset$のとき.
  $p\backslash\mathcal{D}(A;B) = p\backslash\mathcal{D}(A).\mathcal{D}(B) \cup (\mathcal{D}(A)\backslash p)\backslash\mathcal{D}(B)$ だから $p\backslash\mathcal{D}(A).\mathcal{D}(B) \neq \emptyset$ の場合と $p\backslash\mathcal{D}(A).\mathcal{D}(B) = \emptyset \wedge (\mathcal{D}(A)\backslash p)\backslash\mathcal{D}(B) \neq \emptyset$の場合を考えればよい.
  - $p\backslash\mathcal{D}(A).\mathcal{D}(B) \neq \emptyset$の場合.
    $\mathcal{D}(B) \neq \emptyset$なので$p\backslash\mathcal{D}(A) \neq \emptyset$. 帰納法の仮定より $\exists C$ *s.t.* $\langle A,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$. 遷移規則から $\langle A;B,\sigma\rangle \xrightarrow{p} \langle C;B,p(\sigma)\rangle$. よって$p.\mathcal{D}(C;B) = p.\mathcal{D}(C).\mathcal{D}(B) \subseteq \mathcal{D}(A).\mathcal{D}(B) = \mathcal{D}(A;B)$.
  - $p\backslash\mathcal{D}(A).\mathcal{D}(B) = \emptyset \wedge (\mathcal{D}(A)\backslash p)\backslash\mathcal{D}(B) \neq \emptyset$ の場合.
    $\mathcal{D}(B) \neq \emptyset$だから$p\backslash\mathcal{D}(A) = \emptyset \wedge \mathcal{D}(A)\backslash p \neq \emptyset$.
    *derivation*の定義により$\mathcal{D}(A)\backslash p \subseteq \{\varepsilon,p\}$であるが, もし$\varepsilon \in \mathcal{D}(A)\backslash p$とすると $p \in \mathcal{D}(A)$となり $p\backslash\mathcal{D}(A) = \emptyset$に反する. よって $\mathcal{D}(A)\backslash p = \{p\}$.
    つまり$\varepsilon \in \mathcal{D}(A)$であるから定理*14*により $\langle A,\sigma\rangle \xrightarrow{\varepsilon} \langle E,\sigma\rangle$.
    一方 $(\mathcal{D}(A)\backslash p)\backslash\mathcal{D}(B) \neq \emptyset$であるが上記の議論により$p\backslash\mathcal{D}(B) \neq \emptyset$である. よって 仮定から$\exists C$ *s.t.* $\langle B,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$. 遷移規則から $\langle A;B,\sigma\rangle \xrightarrow{\varepsilon} \langle E;B,\sigma\rangle \xrightarrow{\varepsilon} \langle B,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$.
    よって $\langle A;B,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$.
    また $p.\mathcal{D}(C) \subseteq \mathcal{D}(B) = \varepsilon.\mathcal{D}(B) \subseteq \mathcal{D}(A).\mathcal{D}(B) = \mathcal{D}(A;B)$.
- $p\backslash\mathcal{D}(A+B) \neq \emptyset$の場合.
  $p\backslash\mathcal{D}(A+B) = p\backslash\mathcal{D}(A) \cup p\backslash\mathcal{D}(B)$なので$p\backslash\mathcal{D}(A) \neq \emptyset$の場合を考えれば十分. 仮定から $\exists C$ *s.t.* $\langle A,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$. 遷移規則から $\langle A+B,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$. よって $\langle A+B,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$.
  また $p.\mathcal{D}(C) \subseteq \mathcal{D}(A) \subseteq \mathcal{D}(A) \cup \mathcal{D}(B) = \mathcal{D}(A+B)$.
- $p\backslash\mathcal{D}(A\|B) \neq \emptyset$の場合.
  $p\backslash\mathcal{D}(A\|B) = (p\backslash\mathcal{D}(A))|\mathcal{D}(B) \cup \mathcal{D}(A)|(p\backslash\mathcal{D}(B))$ なので$p\backslash\mathcal{D}(A) \neq \emptyset$の場合を考えれば十分. 仮定から $\exists C$ *s.t.* $\langle A,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$. 遷移規則から$\langle A\|B,\sigma\rangle \xrightarrow{p} \langle C\|B,p(\sigma)\rangle$. また $p.\mathcal{D}(C\|B) = p.(\mathcal{D}(C)|\mathcal{D}(B)) \subseteq \mathcal{D}(A)|\mathcal{D}(B) = \mathcal{D}(A\|B)$ ■

証明 (定理 16)

- $p = \varepsilon$の場合.
  1. $\langle E;A,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle$,
  2. $\langle A+B,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle$,
  3. $\langle A+B,\sigma\rangle \xrightarrow{\varepsilon} \langle B,\sigma\rangle$,
  4. $\langle A\|E,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle$,
  5. $\langle E\|A,\sigma\rangle \xrightarrow{\varepsilon} \langle A,\sigma\rangle$.

  以上がすべての遷移であり そのどの場合も明らかに命題を満たしている.
- $p \in \mathcal{F}$の場合.
  $\mathcal{F}$の定義から$\exists c \in \mathcal{C}$ *s.t.* $p = f[c]$である.
  Aに関する構造帰納法で証明する.
  - $A = c$のときは自明.
  - $A;B$のとき.
    遷移規則から$\exists C$ *s.t.* $\langle A;B,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$ であれば$\exists A'$ *s.t.* $\langle A,\sigma\rangle \xrightarrow{p} \langle A',p(\sigma)\rangle$ かつ $C = A';B$ である. ここで帰納法の仮定から$p.\mathcal{D}(A') \subseteq \mathcal{D}(A)$. よって $p.\mathcal{D}(A';B) = p.\mathcal{D}(A').\mathcal{D}(B) \subseteq \mathcal{D}(A).\mathcal{D}(B) = \mathcal{D}(A;B)$.
  - $A\|B$のとき.
    遷移規則から$\exists C$ *s.t.* $\langle A\|B,\sigma\rangle \xrightarrow{p} \langle C,p(\sigma)\rangle$ であれば$\exists A'$ *s.t.* $\langle A,\sigma\rangle \xrightarrow{p} \langle A',p(\sigma)\rangle$ かつ $C = A'\|B$,
    もしくは$\exists B'$ *s.t.* $\langle B,\sigma\rangle \xrightarrow{p} \langle B',p(\sigma)\rangle$ かつ $C = A\|B'$である.
    しかし前者の場合だけ考えれば十分である. ここで仮定から $p.\mathcal{D}(A') \subseteq \mathcal{D}(A)$. よって $p.\mathcal{D}(A'\|B) = p.(\mathcal{D}(A')|\mathcal{D}(B)) \subseteq \mathcal{D}(A)|\mathcal{D}(B) = \mathcal{D}(A\|B)$. ■

証明 (定理 17)

(略) ■

5. 安全性解析

意味論$\mathcal{D}$を用いると初期状態$\sigma \neq \perp$においてプログラムAがタプルの過剰削除によるエラーを起こすことは $(\sigma,\perp) \in \lceil\mathcal{D}(A)\rceil$ と表せる. 逆に $(\sigma,\perp) \notin \lceil\mathcal{D}(A)\rceil$ であるときに初期状態σにおいてプログラムAは過剰削除を起こさない. 同様のことが過剰追加についてもいえる. ここで任意の初期状態に対して上記のエラーを起こさないということを定義する.

定義 18 (安全性) 任意の初期状態に対して過剰削除や過剰追加を起こさないことを安全であると呼ぶ. つまり, $f \in \mathcal{F}$, $\vec{\alpha} \in \mathcal{F}^*$, $X \in \wp(\mathcal{F}^*)$ たちはそれぞれ以下の条件を満たすときに安全であると呼ぶ.

- $f \sqcap \mathbf{D} = \emptyset$.
- $\lceil\vec{\alpha}\rceil \sqcap \mathbf{D} = \emptyset$.
- $\lceil X\rceil \sqcap \mathbf{D} = \emptyset$.

ここで, $\mathcal{S}$上の関係$\mathbf{D}$を, タプル空間が無限の場合は
$\mathbf{D} := \{(\sigma,\perp) \in \mathcal{S} \times \mathcal{S} | \sigma \neq \perp\}$とし,
タプル空間が有限の場合は
$\mathbf{D} := \{(\sigma,\perp),(\sigma,\top) \in \mathcal{S} \times \mathcal{S} | \sigma \neq \perp, \sigma \neq \top\}$とした.

事実 19 以下の性質が成り立つ.

- εは安全である.
- $\vec{\alpha},\vec{\beta}$が安全ならば$\vec{\alpha}.\vec{\beta}$, $\vec{\alpha}+\vec{\beta}$は安全である.
- $\vec{\alpha}.\beta$が安全ならば$\vec{\alpha}$は安全である.

直感的には安全なもの同士を合成すれば安全なものが出来るだろうと考える．実際 逐次合成と選択においてはそれは真実である．しかし 並列合成に関してはその直感は正しくない．以下の命題の反例を挙げることでそれを示す．

命題 20 $\vec{\alpha}, \vec{\beta}$が安全ならば$\vec{\alpha}|\vec{\beta}$は安全である．

反例 1

$\alpha = \{(\bot,\bot),(1,2),(2,2),(\top,\top)\}$

$\beta = \{(\bot,\bot),(1,\bot),(2,2),(\top,\top)\}$

$\gamma = \{(\bot,\bot),(1,2),(2,1),(\top,\top)\}$とし$(\alpha.\beta)|\gamma$を考える．

γは安全である．$\alpha\beta = \{(\bot,\bot),(1,2),(2,2),(\top,\top)\}$であるので$\alpha.\beta$も安全である．しかし$\alpha.\gamma.\beta \in (\alpha.\beta)|\gamma$は$\alpha\gamma\beta = \{(\bot,\bot),(\mathbf{1},\bot),(\mathbf{2},\bot),(\top,\top)\}$であるので$(\alpha.\beta)|\gamma$は安全ではない．

ただ この反例は一般的な協調実行言語とは様子が異なるように思える．実際 例3に挙げたようなタプル空間操作を用いる場合は安全なもの同士の並列合成も安全となる．この違いはタプル空間操作の性質によるものである．上の例では α, β は単調増加関数 $(\sigma \subset \tau \Rightarrow \alpha(\sigma) \subseteq \alpha(\tau)$．便宜上 $\bot \subset \emptyset, \forall\sigma \neq \top.\sigma \subset \top.)$ だが γ は単調増加関数ではない．それに対し 例3の操作関数はすべて単調増加関数である．よって タプル空間操作関数がすべて単調増加関数であれば命題20は成り立つように思える．しかし リセット関数 $f[reset] = \{(\sigma,\emptyset)|\sigma \neq \bot\} \cup \{(\bot,\bot)\}$ も安全であるという事実からその命題には以下のような反例が作れる．

反例 2

$f[put(t)].f[get(t)]$ は $\lceil f[put(t)].f[get(t)]\rceil = id$ であるので安全である．しかし $f[reset]$ との並列合成 $(f[put(t)].f[get(t)])|f[reset]$ にはすべての状態から $\bot$ に陥る$f[put(t)].f[reset].f[get(t)]$というパスが存在する $(\forall\sigma.\lceil f[put(t)].f[reset].f[get(t)]\rceil(\sigma) = \bot)$.

全て安全であるような集合であれば命題20は成り立つがそれではエラーチェック機構としての$\bot$,$\top$の存在意義がない．以上のことを踏まえて命題20を満たすようなタプル空間操作関数の集合$\mathcal{F}$の意味のある条件を求めることが今後の課題である．

6. ま と め

関係文字列の概念を提案することで協調実行言語の意味を比較的簡単な表現で与えることができた．しかし 本論文で用いたシステムは再帰を持たない，排他的な読み書きを行う，という2つの非常に強い制限のかけられたシステムであったからこそ簡単な表現で意味を与えることができたのである．

現実のシステムでは再帰はごく普通に存在するが今回の意味論に単純に再帰を追加すると 意味集合は無限列の無限集合となり有効な解析ができなくなる．

並列計算機にはいくつかの形態があるが複数のPCをネットワーク接続したクラスタシステム(PCクラスタ)はコストパフォーマンスの面などから今後一層普及していくと思われる．しかし PCクラスタ上で(物理的には分散された)共有空間を実現したときに"共有空間へのアクセスは排他的に行う"という制限はシステムの足枷となり性能低下をもたらすことは明白である．

今後の課題は上記の2つの強い制限を取り払ったシステムに対して有効な解析を行える意味論を構成することである．

参 考 文 献

1) Antonio Brogi and Jean-Marie Jacquet, Modeling Coordination via Asynchronous Communication, LNCS 1282, pp.238–255, 1997.
2) Baeten, J.C.M. and Weijland, W.P., Process Algebra, Cambridge University Press, 1990.
3) David Gelernter, Generative Communication in Linda, ACM TOPLAS, Vol.7-1, pp.80–112, 1985.
4) De Nicola R. and Pugliese R., Testing semantics of asynchronous distributed programs, LOMAPS 1996, pp.320–344, 1996.
5) Hoare, C.A.R., Communicating Sequential Process, Pearson Academic Computing, 1986.
6) Jules Desharnais and Bernhard Möller, Characterizing Functions in Kleene Algebras, RelMiCS 2000, pp.55–64, 2000.
7) Robin Milner, Communication and Concurrency, Prentice Hall, 1989.