

CCS semanticsをベースとした

通信システムの記述法

Communicating Systems Description Method Based on CCS semantics

村上 龍郎

Tatsurou MURAKAMI

N T T 電気通信研究所

N T T Electrical Communications Laboratories

あらまし 通信システムのための2種の異なる記述モデルについて述べる。

一方は、ユーザサイドに立ってシステム全体を観測的に記述するサービス記述のモデルで、他方はシステム各部の動作やそれらの間のプロトコルを記述するシステム記述のモデルである。これらのモデルは対なる表現形態と見なすことができ、その動作の意味はCCS semanticsをベースに与えた。また、両モデル間の相互変換の規則を定義した。このとき変換の前後でCCSの"observation congruence"が保証される。

Abstract This paper describes two different description models of communicating systems. One is for service description which describes the observable behaviors of a whole system from user's standpoint and the other is for system description which describes the behaviors of system elements and the protocol between the elements. These description models can be regarded as a dual system. Their semantics are given based on CCS semantics. This paper also presents the mutual transformation methods which keep "observation congruence" between original descriptions and the translated descriptions.

1. まえがき

通信システムは並行動作する複数の制御要素から成っていること、及びシステム全体の動作自体に並列性があることに特徴がある。

このような通信システムを記述する場合、システム的设计段階によって、①システム全体の動作のユーザサイドから見た記述(以降、この記述を"サービス記述"と呼ぶ)や②各制御要素に対する要求動作や制御要素間のプロトコルの記述(以降、この記述を"システム記述"と呼ぶ)が必要とされている。^{(1)・(2)}

二つの記述のうちシステム記述に関しては様々なモデル⁽³⁾や記述言語^{(4)・(5)・(6)}が提案されており、実際これらを使用して記述されている。一方、サービス記述に関しては、自然言語^{(1)・(7)}で記述されたり、システム記述のためのモデルや記述言語を使用して記述されている。^{(4)・(6)}自然言語で記述した場合は形式性に欠ける。また、システム記述と同じ表現形態で記述した場合は両記述の関係が導き易い利点があるが、サービス記述とシステム記述の間の性格の違いを反映できないため、サービス記述はいまいちな表現になったり、必要以上の情報が記述され、過度に複雑な表現

になったりする。

そこで本検討では、サービス記述とシステム記述の特徴①、②を考慮して、それぞれの記述に適した表現形態を提案する。次に、各記述の表す動作(意味)を並行プログラムの理論であるCCS (A Calculus of Communicating Systems)⁽⁶⁾と同様の形式で与える。さらに両記述間の相互変換に関し、動作の同等性を保証する変換規則を示す。

以降、2章では、サービス記述、システム記述に対応する表現の基本的特徴を、3章では、両表現のシンタックスとセマンティクスを、4章では、両表現間の交換法と交換されるための条件について述べる。

2. alternative と concurrencyに着目した二種類の表現法 (A D E & C D E)

通信システムの記述モデルを導き出すために、通信システムの動作の特徴である"alternative"と"concurrency"の二つに注目する。

ここで"alternative"とは選択可能な動作のうちどの動作が選ばれるかという分岐を指し、"concurrency"とは動作順に因果関係があるか独立かという並列性を

指す。これをCCS表現で考えると"alternative"はオペレータ"+"に、concurrencyはオペレータ"|"に対応しており、

$P \xrightarrow{a} P'$, $P \xrightarrow{b} P''$ なる動作に対して、
 alternative に着目した表現は $P = a P' + b P''$
 concurrency に着目した表現は $P = (a q_1) | (b q_2)$
 (但し、 $P' = q_1 | (b q_2)$, $P'' = (a q_1) | q_2$)
 となる。但し、 $P \xrightarrow{a} P'$ は "a という動作の結果、プログラムP がP'になる"ことを表す。

さて、通信システムの最も代表的な表現法は、独立に動作する動作実体(一般に"プロセス"と呼ばれている)を $P_1 || P_2 || \dots || P_n$ (P_i はプロセス)のように複数定義することによって、動作のconcurrencyをプロセスの独立な動作とその間の同期信号を通して表現し、動作のalternativeを各プロセス P_i 内に直接表現する。通信システムやプロトコルの仕様記述言語として使われているSDL⁽⁴⁾、Estelle⁽⁵⁾、CSP⁽⁶⁾はこの表現法に基づいている。本検討では alternative と concurrencyを対なる関係と見なすことによって、上記表現形態に相対な表現形態を考える。この表現形態は、 $Q_1 \# Q_2 \# \dots \# Q_m$ のように動作のalternativeは選択的動作を表す Q_i の組で表現し、concurrencyは Q_i 内に直接表現する。ここで P_i をプロセスと呼ぶのに対応して、 Q_i をBインスタンス(Behaviorインスタンス)と呼ぶ。また、 P_i の表現形態ではプロセス間の動作の順序関係(同期関係)を同期信号で規定するのに対して、 Q_i の表現形態ではBインスタンス間の動作の分岐関係を規定する記述が必要である。両表現形態の相対イメージを図1に示した。本検討では、プロセス内に alternativeを直接表現した表現形態を"ADE (Alternative Directed Expression)"と呼び、これに對なるBインスタンス内にconcurrencyを直接表現した表現形態を"CDE (Concurrency Directed Expression)"と呼ぶ。

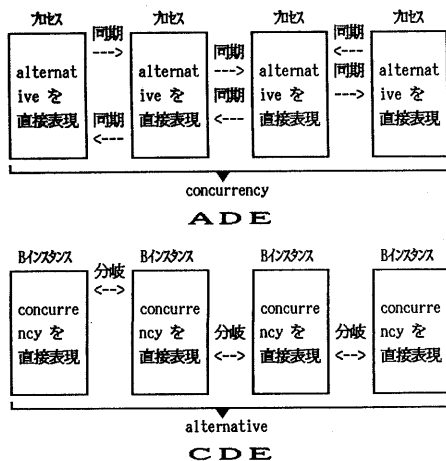


図1. ADE&CDEのイメージ

ADE & CDEを通信システム記述への適用という点から考えると、ADEは従来からシステム記述に使用されている仕様記述言語の記述形態である。一方、CDEは記述単位が一回の動作の実現単位と対応していること、システム外部に現われる動作の並列性をそのまま表現していることからシステムの外部動作を記述するサービス記述の表現形態に向く。また、alternativeを組に分けて記述するのでサービス記述に要求されるサービスの追加の扱いが容易になる。CDEの記述単位 $Q_1 \# Q_2 \# \dots \# Q_m$ と記述単位が類似しているメッセージシーケンスチャートや仕様記述言語SAL⁽⁹⁾は実際、通信システムのサービスに近い部分の記述に使われている。

ADE & CDEの動作の定義は、並行プログラムの一モデルとして広く知られているCCSのセマンティクスに對照させる。ここでCCSを選んだ理由はCCSにおいてシステムの動作にobservationalな観点から意味を与えている点が、システム各部の動作を表すシステム記述とユーザサイドから見た外部動作を表すサービス記述との間の対応を与える本検討の目的に適合しているためである。

CCSではconcurrency"|"と alternative"+"の間には $a|b = ab+ba$ なる意味付け(Expansion Theorem)がなされている。そして、外部動作の意味には、外部に現われる並列性は露わに表現されない(外部動作の代表的モデルはconcurrencyがalternativeの形に展開されたツリー(communication tree)である)。これはCCSのセマンティクスが外部からの(因果関係や独立性を見分けられない)シリアルな観測を基本としていることによる。

しかしながら、本検討でCDEを適用するサービス記述では観測現象間の因果関係や独立性を表現するため、 $a|b$ と $ab+ba$ が意味的に異なるという立場をとる。そこでADE & CDEにおける動作の同等性は、CCSで定義されている(展開された木が同等であるという意味の)観測上の同等性(observation congruence)のみならず、concurrencyが展開されず保存されていることを条件に定義する。(これによってconcurrencyがalternativeの形に展開されたものは"同等"から排除される)

3. シンタクスとセマティクス

ADEはCCSのサブセットとしてシンタクスとセマンティクスを与える。CDEはシンタクスを新たに定義し、セマンティクスはCCSと同様な方法で定義する。これによって、ADEとCDEの動作が比較可能となる。

ADEは $P_1 || P_2 || \dots || P_n$ のようにプロセス P_i の組で表す。各 P_i はイベントとオペレータ "+", "(", ")" からなる式であり、動作の分岐関係はツリーである。

CDEは $Q_1 \# Q_2 \# \dots \# Q_m$ のようにBインスタンス Q_i の組で表す。各 Q_i はイベントとオペレータ "<=", "(", ">" からなる式であり、"<="は因果関係を表し、"(", ">"は

その並置を表す ($e_1 \ll e_2$ は「 e_1 の後に e_2 が起こる」を意味し、 e_1 と e_2 に順序がない は「 e_1 と e_2 の間に因果関係がない」を意味する)。動作の因果関係は半順序集合 (Hasse 図形に対応) である。

3.1 ADE

【領域】

Σ : イベント (動作の単位) の集合, $\Sigma = \Delta + \bar{\Delta}$,
(For $\forall \lambda \in \Sigma, \bar{\lambda} \in \Sigma$ and $((\lambda \in \Delta \text{ iff } \bar{\lambda} \in \bar{\Delta})$
or $(\lambda \in \bar{\Delta} \text{ iff } \bar{\lambda} \in \Delta))$ and $\bar{\bar{\lambda}} = \lambda$)

$\Sigma_1 = \Sigma \cup \{\tau\}$ (τ は観測されない要素)

【CCSシンタクス

(rename と value passing を除く)】⁽⁸⁾

e: 動作識別ラベル

E: 動作表現式

"+", "|", "\", "\ " は CCS のオペレータ

$\lambda \in \Sigma, a \in \Delta$ とする

$e \Leftarrow E$ (動作識別ラベルと動作表現式の対応)

$E ::= \text{NIL} \mid e \mid \lambda E \mid \tau E \mid E + E' \mid E|E' \mid E \setminus a$ □
次のような関数を定義する。

L, L_1 : 動作表現中のイベントを抽出する関数で、以下のように定義する。但し、 $a \in \Sigma$

$L(E) = s$ ($s \subset \Sigma$), $L(\text{NIL}) = \phi$, $L(aE) = L(E) \cup \{a\}$
 $L(e) = L(E)$, $L(\tau E) = L(E)$, $L(E \setminus a) = L(E) - \{a, \bar{a}\}$,
 $L(E + E') = L(E) \cup L(E')$, $L(E|E') = L(E) \cup L(E')$,
 $L_1(e) = \phi$, それ以外は L と L_1 は同じ。

names: イベントの "-" を取り去る関数。

任意の $s \subset \Sigma$ に対して, $\text{names}(s) = (s \cup \bar{s}) \cap \Delta$
また、次の省略形を定義する。

$E \setminus a_1 \setminus a_2 \setminus \dots \setminus a_n = E \setminus A$, $A = \{a_1, a_2, \dots, a_n\}$
 $(E_1|E_2) \setminus A = E_1|E_2$ ($A = \text{names}(L(E_1) \cap L(E_2))$) □

【ADEシンタクス (CCSのサブセット)】

p: 動作識別ラベル (定義した動作識別ラベル)

P: 動作表現式 の集合を Ω_p とする

$\lambda \in \Sigma$ とする。

$p \Leftarrow P$ (動作識別ラベルと動作表現式の対応)

$P ::= P_1 \parallel P_2 \parallel \dots \parallel P_n$, n: 有限整数 (P_i : プロセス)

$P_i ::= \text{NIL} \mid p \mid \lambda P_i \mid \tau P_i \mid P_i + P_i'$

(a) $L_1(P_i) \cap L_1(P_j) = \phi$ ($i \neq j$)

(b) P_i 中の任意の p に対して $L(p) \cap L(P_j) = \phi$ ($i \neq j$)

(c) 動作表現式内に現れる $p \in \Omega_p$ は Σ_1 の要素によってガードされている。(文献(8)の5.4に述べられている "guardedly well-defined" である) □
制約 (a), (b) から、各プロセス間の同期を表すイベント対のラベルがユニークであることが言える。

$1 \leq \forall i, j, k \leq n$ に対して

$\text{names}(L(P_k) \cap L(P_i)) \cap \text{names}(L(P_k) \cap L(P_j)) = \phi$

【ADEセマンティクス】

文献(8)から次のように導かれる。

$\xrightarrow{\mu}$ は各 $\mu \in \Sigma_1$ について定義された ADE プログラム間の二項関係で、 $P \xrightarrow{\mu} P'$ は、" μ という動作の結果、ADE プログラム P は P' になる" を表す。

(1) NIL has no action.

(2) $\mu P_1 \xrightarrow{\mu} P_1$

(3) $\frac{P_1 \xrightarrow{\mu} P_1'}{P_1 + P_2 \xrightarrow{\mu} P_1'} \quad \frac{P_2 \xrightarrow{\mu} P_2'}{P_1 + P_2 \xrightarrow{\mu} P_2'}$

for $1 \leq \forall i \leq n, \forall \mu \notin A$ ($A = \text{name}(\cup_{j=1}^n (L(P_j) \cap L(P_k)))$),

(4) $\frac{P_1 \xrightarrow{\mu} P_1'}{P_1 \parallel P_2 \parallel \dots \parallel P_n \xrightarrow{\mu} P_1' \parallel P_2 \parallel \dots \parallel P_n}$

(5) $\frac{\text{for } 1 \leq \forall i \leq n, P_i \xrightarrow{\lambda} P_i' \text{ and } P_j \xrightarrow{\bar{\lambda}} P_j'}{P_1 \parallel \dots \parallel P_n \xrightarrow{\tau} P_1' \parallel \dots \parallel P_n'}$

(6) $\frac{p \Leftarrow P, P \xrightarrow{\mu} P'}{p \xrightarrow{\mu} P'}$

3.2 CDE

CDE は動作の alternative を $Q_1 \# Q_2 \# \dots \# Q_m$ のように B インスタンスの組で表す。そこで、図 2②の動作は CDE では ②' のように表す。②' が ②, ③' ではなく ② を表現するためにはシステム動作の分岐は異なるイベントで起こることが要請される。②, ③' のような非決定的動作に対しては、観測されない要素の集合 G, H を領域に加えてこの要素を用いて表現する (②' のような動作に対しては $g_1, g_2 \in G$ を付与して ②' のように、③' のような動作に対しては $h_1 \in H$ を付与して ③' のように表す)。以上により、異なるイベントでシステム動作が分岐するという形式化がなされる。以降、 G, H の要素を "分岐因子" と呼ぶ。

CDE の B インスタンスが表す動作はイベントの半順序集合で表現するので、同じラベルで表現されるイベントが 2 回以上起こる場合にこれらの順序関係を表すため半順序集合の要素として区別する必要がある。そこで、イベントの領域を $\Sigma \times N$ に拡張する。

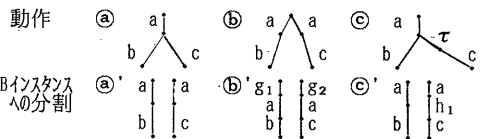


図 2. 非決定的動作の例

【領域】

$\Sigma_2 = \Sigma \cup G \cup H$ (N は自然数, Σ, G, H は
 $\Sigma_3 = \Sigma_2 \times N$ 共通部分を持たない)
簡単のため任意の要素 $(a, i) \in \Sigma_3$ を a^i と表す。
また、 \bar{a}^i は (\bar{a}, i) を意味する。

【CDEシンタクス】

q: 動作識別ラベル (定義した動作識別ラベル)

Q: 動作表現式 の集合を Ω_q とする

"#", "<=", "\", "\ " は CDE のオペレータ

$\alpha, \alpha' \in \Sigma_3, (q, j) \in \Omega_q, \times N$ を q^j と表す。

$q \Leftarrow Q$ (動作識別ラベルと動作表現式の対応)

$Q ::= Q_1 \# Q_2 \# \dots \# Q_m$, m: 有限整数 (Q_i : B インスタンス)

$Q_i ::= \{ R \}$

$R ::= \text{NIL} \mid q^j \mid \alpha \Leftarrow \text{NIL} \mid \alpha \Leftarrow q^j \mid q^j, R \mid$

$\alpha \Leftarrow \text{NIL}, R \mid \alpha \Leftarrow q^j, R \mid \alpha \Leftarrow \alpha', R$

・各 Q_i は次の(a) ~ (c)の条件を満たさなくてはならない。各 Q_i に対して関数 W と順序関係 \leq_i を定義する
 W : 任意の B インスタンスに対して, R に現れる Σ_3 ,
 $\Omega_q \times N$ の要素, NIL を集合として抽出する関数

$$W(Q_i) \subset \Sigma_3 \cup (\Omega_q \times N) \cup \{NIL\}$$

\leq_i : $W(Q_i)$ の要素間の二項関係で次のように定義する

$$Q_i \text{ が } \{ \dots, a_1 \leq a_m, \dots \} \text{ なら } a_1 \leq_i a_m$$

$$a_1 \leq_i a_m \text{ かつ } a_m \leq_i a_n \text{ なら } a_1 \leq_i a_n$$

各 Q_i は次の条件を満たしなくてはならない。

$$(a) \forall a_1, a_m \in W(Q_i) \text{ で } a_1 \leq_i a_m \text{ なら } a_m \leq_i a_1$$

$$(b) \forall a_1 \in W(Q_i) \text{ で, } a_1 \leq_i a_1$$

$$(c) \forall a_1 \in W(Q_i) \cap \Sigma_3 \text{ で, } a_1 \leq_i NIL \text{ か } a_1 \leq_i q^j \quad \square$$

このとき Q_i に対して半順序集合 $(W(Q_i), \leq_i)$ が一意的に決定される。(4章では, $Q_i = \{R\}$ の形の表現より対応させた半順序集合 $(W(Q_i), \leq_i)$ の表現を主に使う)

以下の議論のために任意の半順序集合 $\xi = (W, \leq)$ に対して次の定義をする。

first: ξ の最も順序の小さい要素を抽出する関数。

$$\text{first}(\xi) = \{a_1 \in W \mid \text{for } \forall a_m \in W \ a_m \leq_i a_1\}$$

last: ξ の最も順序の大きい要素を抽出する関数。

$$\text{last}(\xi) = \{a_1 \in W \mid \text{for } \forall a_m \in W \ a_1 \leq_i a_m\}$$

\leq^d : $a \leq^d b$ とは, $a \leq b$ で, $a \leq c \leq b$ なる c が存在しないことである

【CDEセマンティクス】

$\xrightarrow{\mu}$ をCDEプログラム間の二項関係にまで定義を広げる。

(1) $\{NIL\}$ has no action.

$$(2) \frac{\alpha \in \text{first}(Q_i) \cap \Sigma_3}{Q_i \xrightarrow{\alpha} Q_i', W(Q_i') = W(Q_i) - \{\alpha\} \text{ and } (\text{for } \forall a_j, a_k \in W(Q_i'), a_j \leq_i' a_k \text{ iff } a_j \leq_i a_k \text{ and } \leq_i \text{ は } Q_i \text{ の, } \leq_i' \text{ は } Q_i' \text{ の順序関係})}$$

for $\forall a^i \in \Sigma \times N, \exists J \subset \{1, \dots, m\} (J = \{j_1, \dots, j_n\}, \underline{J} = \{1, \dots, m\} - J)$

for $\forall j \in J, Q_j \xrightarrow{a^j} Q_j'$ and for $\forall k \in \underline{J}, Q_k \xrightarrow{a^k} Q_k'$

$$(3) \frac{Q_1 \# Q_2 \# \dots \# Q_m \xrightarrow{a} Q_1' \# Q_2' \# \dots \# Q_m'}{\text{for } \forall g^p \in G \times N, \forall a^i \in \Sigma \times N, \exists J \subset \{1, \dots, m\} (J = \{j_1, \dots, j_n\}, \underline{J} = \{1, \dots, m\} - J)}$$

for $\forall j \in J, Q_j \xrightarrow{g^p} Q_j'$ and $g^p \leq_i a^i$ and $g^p \leq_i a^i$

$$(4) \frac{\text{for } \forall k \in \underline{J} \quad Q_k \xrightarrow{g^p} Q_k'}{Q_1 \# Q_2 \# \dots \# Q_m \xrightarrow{a} Q_1' \# Q_2' \# \dots \# Q_m'}$$

for $\forall h^p \in H \times N, \exists J \subset \{1, \dots, m\} (J = \{j_1, \dots, j_n\}, \underline{J} = \{1, \dots, m\} - J)$

$$(5) \frac{\text{for } \forall j \in J, Q_j \xrightarrow{h^p} Q_j' \text{ and for } \forall k \in \underline{J}, Q_k \xrightarrow{h^p} Q_k'}{Q_1 \# Q_2 \# \dots \# Q_m \xrightarrow{h^p} Q_1' \# Q_2' \# \dots \# Q_m'}$$

for $\forall q^i \in \Omega_q \times N, \exists J \subset \{1, \dots, m\} (J = \{j_1, \dots, j_n\}, \underline{J} = \{1, \dots, m\} - J)$

for $\forall j \in J, q^i \in \text{first}(Q_j)$ and for $\forall k \in \underline{J}, q^i \notin \text{first}(Q_k)$

and $q^i \leq_i q^i$ and $Q \xrightarrow{a} Q'$ ($Q' = Q_1' \# \dots \# Q_n', a \in \Sigma$)

$$(6) \frac{\text{for } \forall j \in J, Q_j \xrightarrow{a^{n+1}} Q_j', q^i = Q_1' \# \dots \# Q_n', \text{ for } 1 \leq i \leq n, W(Q_i') = (W(Q_i) - \{q^i\}) \cup W'(Q_i') \text{ (for } \forall b^p \in (\Sigma_2 \cup \Omega_q) \times N, b^p \leq_i' W'(Q_i') \text{ iff } b^p \leq_i W(Q_i')) \text{ and (for } \forall \alpha, \beta \in (W(Q_i) - \{q^i\}), \alpha \leq_i' \beta \text{ iff } \alpha \leq_i \beta \text{ and for } \forall b^p, c^p \in W(Q_i'), b^p \leq_i' c^p \text{ iff } b^p \leq_i c^p \text{ and } \leq_i \text{ は } Q_i \text{ の, } \leq_i' \text{ は } Q_i' \text{ の, } \leq_i'' \text{ は } Q_i'' \text{ の順序関係) 但し, } n = \max\{x \mid \text{for } \forall j \in J, \forall b^p \in W(Q_j) \text{ の } N \text{ の射影}\}}{Q_1 \# Q_2 \# \dots \# Q_m \xrightarrow{a^{n+1}} Q_1' \# Q_2' \# \dots \# Q_m'}$$

4. 両記述間の相互変換

ADE, CDEの記述間の相互変換規則について述べる。変換に際して次のことを条件とする。

【変換に対する条件】

(1) 変換前後の記述(対応するADEプログラムとCDEプログラム)はobservation congruent⁽⁶⁾である

(2) 変換の前後で並列性は保存される(付録参照) □

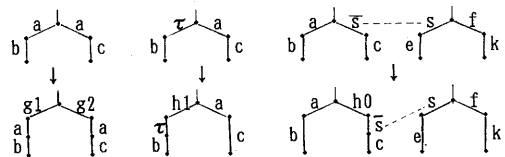
なお, 本章ではADE, CDEの各プロセス, Bインスタンスを簡単のために, それぞれ tree(アークにイベントを対応させる), Hasse図形(ノードにイベントを対応させ, 順序は下方で矢印は略す)で表す。

4.1 ADEからCDEへの変換

ADEの各プロセスのトレースの組がCDEのBインスタンスに対応する。しかし, 非決定的な動作に対しては, 3.2に示したようにそのままトレースをとって変換したのでは動作を再現できず, observation congruentなものに変換されない。⁽¹⁰⁾そこで, 非決定的動作が生じる部分に対して分岐因子を付与する。

【変換規則】

① ADEの非決定的な動作を生じる分岐に対して, トレースをとった後も動作を再現するように分岐因子をつける。(詳細は付録2-1参照)



但し, $g_1, g_2 \in G, h_1 \in H^1, h_0 \in H^0$

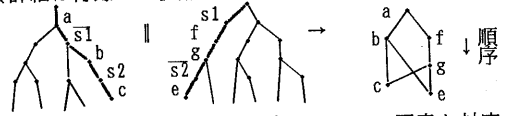
・付与する H, G の要素は全てユニークである。

・便宜的に, $H = H^0 \cup H^1$ かつ $H^0 \cap H^1 = \emptyset$

とし, H^1 の要素は τ の前に, H^0 の要素は同期イベントの前に付与する。

② 分岐因子を付与した各プロセスの分岐の選択子ごとに順序関係を調べて, 半順序集合を生成する。

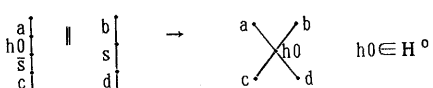
(詳細は付録2-2参照)



・ Σ_3 の要素を $\Sigma_3 \times N$ の要素に, Ω_p の要素を対応する $\Omega_q \times N$ の要素に置き換える。 $(\tau$ は取り除く)

・プロセス間における順序関係は, イベント対の同期の前後のみに付ける。(例えば $(a\bar{s}) \parallel (sb) \rightarrow a \leq b$)

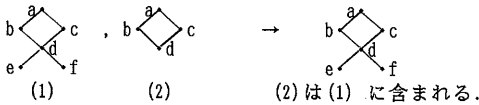
・ H^0 の要素は次のような順序関係に変換する。



・names $(L(P_i) \cap L(P_j))$ の要素で, 同期相手のないイベントが現れた時点(デッドロック)で半順序集合の生成を打ち切る。

③ 生成した半順序集合から, 極大な(他の半順序集合に含まれない)半順序集合を選択する。(詳細は付録

2-3 参照)



・選択した半順序集合 ξ において、
 $\forall a \in (\Sigma_3 \times N) \cap \text{last}(\xi)$ に対して、 $a \ll \text{NIL}$
 なる関係を加える。

上記半順序集合は容易に 3.2 のシンタクスを満たす B
 インスタンス Q_1 に置き換えられ、

CDE プログラム $Q = Q_1 \# \dots \# Q_m$ が得られる。

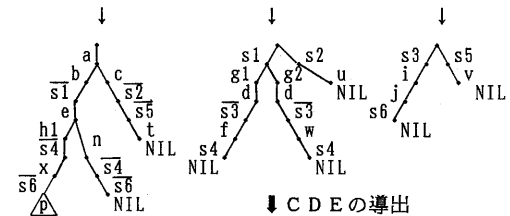
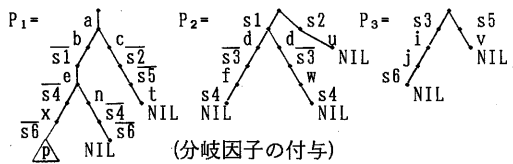
【変換の正当性】

任意の ADE プログラム P に対して、本変換規則で導
 かれる CDE プログラム Q は、 $P \approx^c Q$ (observation
 congruence) かつ 並列性が保存されている。

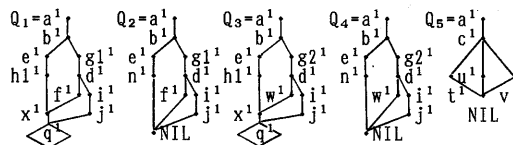
(証明) 省略 (別途報告)

【変換例】

ADE $p \Leftarrow P$ $P = P_1 \parallel P_2 \parallel P_3$



CDE $q \Leftarrow Q$ $Q = Q_1 \# Q_2 \# Q_3 \# Q_4 \# Q_5$



4.2 CDE から ADE への変換

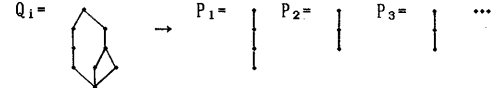
ADE から CDE への変換の場合、生成される B
 インスタンスはユニークに決るが、CDE から ADE へ
 の変換では、各イベントがどのプロセスへの割り当て
 られるかには自由度がある。そこで次の二つのアプ
 ローチが考えられる。

- ① イベントのプロセスへの割り当て情報を変換に際して与える。
- ② プロセス数最小等、所要リソースや実行時間に対する最適化条件を満たす変換方法を求める。
- ③ のアプローチについては、各最適化条件を満たすようなイベントのプロセス割り当てアルゴリズムを与えることによって、①の変換規則と組み合わせて導かれる。この意味で①が基本的であるので、本稿では①のアプローチに対する変換規則を与える。

また、任意の CDE、任意のプロセス割り当てからは、必ずしも先に示した 2 条件を満たす ADE を導くとは限らない。プロセスに振り分けるときにトレースにならなかったり、分岐因子がないのに非決定的分岐になったり、変換される ADE に、もとの CDE がない余分な動作が含まれたりすることがある。そこで、本変換を適用できる CDE の条件 (十分条件) も合わせて示す。

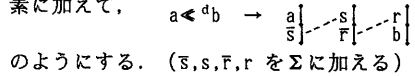
【変換規則】

① 各 B インスタンスからイベントのプロセス割り当てに従ってプロセスのトレースを生成する。(詳細は付録 3-1 参照)

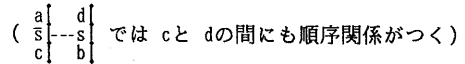


・ $\Sigma_3 \times N$ の要素を Σ_3 の要素へ、 $\Omega_a \times N$ の要素を対応する Ω_p の要素に置き換える。

・ 異なるプロセスに属するイベントの順序関係に対して、同期関係を規定するイベント対を新たに Σ の要素に加えて、



のようにする。(s, s, f, r を Σ に加える)



② 各 B インスタンスから生成したトレースをプロセスごとに合成して tree を生成する。(詳細は付録 3-2 参照)

- ・ 異なるイベントで動作が分岐するように合成する。
- ・ G の要素、同期するイベントの前に付与されている H の要素は、取り除く。
- ・ 上記以外の H の要素は τ に置き換える。

合成された tree を生成する。(詳細は付録 3-2 参照)
 ・ 異なるイベントで動作が分岐するように合成する。
 ・ G の要素、同期するイベントの前に付与されている H の要素は、取り除く。
 ・ 上記以外の H の要素は τ に置き換える。

【変換条件】

CDE プログラムを $Q = Q_1 \# \dots \# Q_m$ とする。

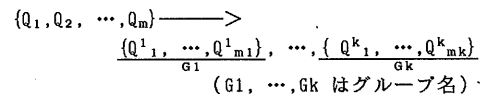
[条件 ①] 各インスタンス Q_i に対して、

$\forall e_1, e_2 \in W(Q_i)$ について、 $\text{process}(e_1) = \text{process}(e_2)$
 なら $e_1 \ll e_2$ か $e_2 \ll e_1$ である。

[条件 ②] $\{Q_1, Q_2, \dots, Q_m\}$ は "分配可能" である。

$\{Q_1, Q_2, \dots, Q_m\}$ が分配可能とは、

$\{Q_1, Q_2, \dots, Q_m\}$ が k 個のグループに以下の条件で分けられて、しかも各グループは分配可能である。



条件: $\forall G_i$ に対して、(1) ~ (4) 満たす $a_i \in \Sigma_3$ (以降、分配点と呼ぶ) が存在する。

(1) $\forall Q_j, Q_k \in G_i$ に対し、 $a_i \in W(Q_j) \cap W(Q_k)$

(2) $\forall G_i, G_j (i \neq j)$ に対して、 $a_i \neq a_j$

(3) $\forall Q_x \in G_i, \forall Q_y \in G_j (i = j \text{ も可})$ に対して

$(e_1, e_2 \in W(Q_x))$ で $e_1, e_2 \prec_x a_1$ かつ $e_1 \prec_x e_2$
 iff $(e_1, e_2 \in W(Q_y))$ で $e_1, e_2 \prec_y a_j$ かつ $e_1 \prec_y e_2$
 (4) $\forall Q_x \in Gi$ に対して, $\exists Q_y \in Gj$,
 $(e_1, e_2 \in W(Q_x))$ で, $a_1 \prec_x e_1, e_2$ かつ $e_1 \prec_x e_2$
 iff $(e_1, e_2 \in W(Q_y))$ で, $a_j \prec_y e_1, e_2$ かつ $e_1 \prec_y e_2$
 (但し, $e_1, e_2 \notin \{a_i, a_j\}$)

[条件③] 一つの分配における各グループの分配点は全て同一プロセスに属す。

$$(Q_1, Q_2, \dots, Q_m) \xrightarrow{\quad} (Q_1^{g_1}, \dots, Q_{m_1}^{g_1}), \dots, (Q_{k_1}^{g_k}, \dots, Q_{m_k}^{g_k})$$

$\forall Gi, Gj$ ($1 \leq i, j \leq k$) に対して, a_i, a_j をそれぞれの分配点とすると $\text{process}(a_i) = \text{process}(a_j)$ □

各条件は次のことを意味する。

③同一プロセスに属する Σ_3 の要素間には必ず順序がつく。

④CDEのBインスタンス間には動作が分岐する部分に相当する Σ_3 の要素(分配点)の対応関係が存在して、一つの対応関係(分岐)は他の対応関係(分岐)に対して因果関係があるか完全に独立かである。

⑤対応する分配点は同一プロセスに属する。

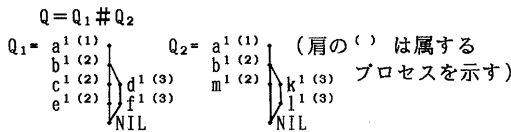
⑥, ⑦はプロセス割り当てに対する条件である。⑥, ⑦は十分条件である。⑥, ⑦を満たさなくてもADEに変換可能なものもある。しかし、このようなものは、「alternativeはプロセス内に表現される」というADEの基本思想からははずれるものである。特に⑥, ⑦の条件を以降、「完全性の条件」と呼ぶ。

【変換の正当性】

変換条件を満たす任意のCDEプログラムQに対して、本変換規則で導かれるADEプログラムPは、
 $P \approx^c Q$ (observation congruence) かつ 並列性が保存されている。

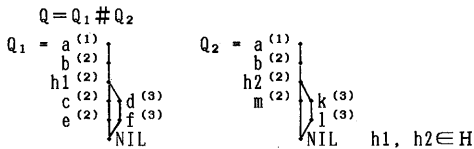
(証明)省略(別途報告)

【変換例】



これは、「完全性の条件」を満たさない。実際、observation congruentなADEは存在しない。しかし、次のような場合CDEは「完全性の条件」を満たす。以降の例では×Nの表示を略す(a^1 をaで示す)。

[case 1] 一つのプロセスにおいて観測可能でない要素で分岐する。



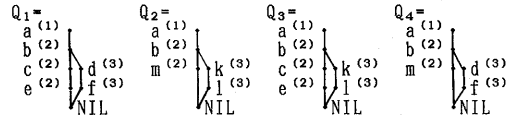
分配点は h_1 と h_2 。↓ ADEの導出

$$P = P_1 \# P_2 \# P_3 \# P_4 \# P_5 \# P_6$$



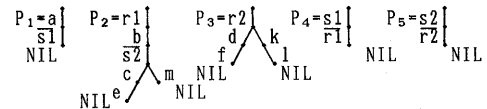
[case 2] プロセス2とプロセス1で独立に分岐する。

$$Q = Q_1 \# Q_2 \# Q_3 \# Q_4$$



分配点は c と m, k と d。↓ ADEの導出

$$P = P_1 \# P_2 \# P_3 \# P_4 \# P_5$$



さて、CDEを外部動作を表すサービス記述、ADEをプロトコルを含んだシステム記述とすると、CDEからADEへの変換は、サービス記述からのプロトコル自動生成に相当する。このような研究は文献(9)や(11)で報告されている。文献(11)はサービス記述からプロトコル表現を排除しているが、両記述を同一言語で記述するため記述範囲が狭くなっている。一方、文献(9)ではサービス記述にサービス向きの記述言語(記述単位はBインスタンスと類似)を用い、変換の際に正当性の検証を行なうが、サービス記述にプロトコル表現が現れる。本検討は、サービス記述を文献(9)と類似のサービス向きの表現形態をとり、しかもそこからプロトコル表現を排除している。

CDEに対する変換可能性条件の中の完全性の条件をプロトコル検証という立場から見ると、「要求外動作の排除」、「プロトコル合成の可能性判定」にあたる。これらの事項はプロトコル記述のレベルでは一般に行われているが⁽¹²⁾、本検討では、(プロトコルを含まない)CDEのレベルで判定する。

5. むすび

通信システムのサービス記述に向く表現形態としてCDEを、システム記述に向く表現形態としてADEを提案した。両表現形態間の関係を導くために、両者を簡単なモデルでシンタクスとセマンティクスを定義し、両者間の相互変換の規則を示した。

特にCDEからADEへの変換は、サービス記述からプロトコルの自動生成に相当する。

なお、今後の課題として次の事項が挙げられる。

(1) ADEとCDEは意味的には対称な表現であるが、本稿で与えたシンタクスはその対称性を明確に表現していない。これは、ADEをCCSのサブセットとして与えたことによる。そこで徹底的に両者の双対性を明らかにする検討が興味深い。

(2) 本検討では、話を簡単にするためにイベントをラベルと見なしそこで引き継がれる値に関しては考慮し

ていない。即ち、CCSのうちvalue-passingを除いた部分を扱った。そこで、次のステップとしてvalue-passingを含めたADE, CDEのシンタクス, セマンティクス, 相互変換規則の拡張を行なう。

(3) 簡単なモデルとして表現したADE & CDEを、具体的な適用を考慮した通信システム記述言語として成長させる。

謝辞 本検討に際し、御助言、御指導頂きました基礎研究所 情報通信基礎研究部 第五研究室 岡田室長、市川主任員、伊藤主任員に深謝致します。

参考文献

- (1) CCITT SG-XVII
- (2) R.Chung: "A Methodology for Protocol Design and Specification based on an Extended State Transition Model", Proc.ACM SIGCOMM Symposium, 14, 2, pp.34-41 (June 1984)
- (3) G.v.Bochmann, C.A.Sunshine: "Formal Methods in Communication Protocol Design", IEEE Trans. Commun., COM-28, 4, pp.624-631 (April 1980)
- (4) CCITT: "Functional Specification and Description Language (SDL)", Recommendations Z.100 ~ Z.104 (1984)
- (5) ISO: "Estelle", ISO/DP9074 (June 1985)
- (6) C.A.R.Hoare: "Communicating Sequential Processes", CACM, Vol.21, 8, pp.666-677 (August 1978)
- (7) 青柳, 長谷川, 田中, 柴田: "通信システムにおける仕様設計エキスパートシステムの一検討", 信学技報 SE86-10 (May 1986)
- (8) R.Milner: "A Calculus of Communicating Systems", Lecture Notes in Computer Science 92 (1980)
- (9) H.Ichikawa, M.Itoh, M.Shibasaki: "Protocol-Oriented Service Specifications and Their Transformation into CCITT Specification and Description Language", Trans.IECE Japan E69, 4, pp.524-535 (April 1986)
- (10) J.Engelfriet: "Determinacy \rightarrow (Observation Equivalence = Trace Equivalence)", Theoretical Computer Science 36, pp.21-25 (1985)
- (11) G.v.Bochmann, R.Gotzein: "Deriving Protocol Specifications from Service Specifications", Proc.of the ACM SIGCOMM'86 Symposium
- (12) P.Zafiropulo, et al.: "Towards Analyzing and Synthesizing Protocols", IEEE Trans.on Commun., COM-28, 4, pp.651-661 (April 1980)

付 録

1. 並列性の保存の定義

なお, " \xrightarrow{S} ", " \approx ", " Σ^* "の定義は文献(8)参照。

【定義】

ADEプログラムPとCDEプログラムQの間で並列性が保存されている iff $\forall \lambda_1, \lambda_2 \in \Sigma, \forall s \in \Sigma^*$ に対して,

- ① $\langle P=P_1 \parallel \dots \parallel P_n \rangle_{\lambda_1 \in L(P_1), \lambda_2 \in L(P_2)}$ で,
 $P_1 \xrightarrow{S} \dots \parallel P_n \xrightarrow{S} \dots \parallel P_1' \parallel \dots \parallel P_j' \parallel \dots$ かつ
 $\dots \parallel P_i' \parallel \dots \parallel P_j' \parallel \dots \xrightarrow{\lambda_1} \dots \parallel P_i'' \parallel \dots$ かつ
 $\dots \parallel P_i' \parallel \dots \parallel P_j' \parallel \dots \xrightarrow{\lambda_2} \dots \parallel P_j'' \parallel \dots$ (状態1) のとき
 $\langle Q=Q_1 \# \dots \# Q_m \rangle$ に対して, $Q_{k_1} \# \dots \# Q_{k_s}$ が存在し,
 $Q_1 \# \dots \# Q_m \xrightarrow{S} Q_{k_1}' \# \dots \# Q_{k_s}'$ かつ
 その中に $\lambda_1, \lambda_2 \in \text{first}(Q_{k_i}')$ なるインスタンス Q_{k_i}' が存在する (即ち, Q_{k_i}' を半順序集合 $(W(Q_{k_i}'), \leq_{k_i})$ に対応させたとき, λ_1, λ_2 の間に順序がない) (状態2)
 かつ $\dots \parallel P_i' \parallel \dots \parallel P_j' \parallel \dots \approx Q_{k_1}' \# \dots \# Q_{k_s}'$ である。
- ② 逆に, $Q=Q_1 \# \dots \# Q_m$ に上記のような $Q_{k_1}' \# \dots \# Q_{k_s}'$ (状態2) があるとき, $P=P_1 \parallel \dots \parallel P_n$ に対して, 上記のような $\dots \parallel P_i' \parallel \dots \parallel P_j' \parallel \dots$ (状態1) が存在して, $\dots \parallel P_i' \parallel \dots \parallel P_j' \parallel \dots \approx Q_{k_1}' \# \dots \# Q_{k_s}'$ である。□

2. ADEからCDEへの変換規則

2-1. 分岐因子付与規則

ADEの $B_1 + \dots + B_n$ なる表現を $\Sigma_i B_i$ と略す。

【定義】 (sumform) B がsumform であるとは, $\Sigma_i g_i B_i$ の形をしており, 各 B_i もsumform の形をしていることを言う。(但し, NIL 動作識別部はsumform である) □

$$H = H^0 \cup H^1 \text{ かつ } H^0 \cap H^1 = \phi$$

【分岐因子付与規則】

ADEプログラムを $P=P_1 \parallel \dots \parallel P_n = (P_1 \parallel \dots \parallel P_n) \setminus A$ とする。

$H = H^0 \cup H^1$ かつ $H^0 \cap H^1 = \phi$ とする。

ここで付与する各 $g_i \in G, h_i \in H$ は全てユニークである。

・各プロセス P_i 内の任意のsumform に対して次の置き換えをする。

- ① $\Sigma_j \lambda B_j$ (但し, $\lambda \in \Sigma$ かつ $j \geq 2$) $\rightarrow \Sigma_j g_j \lambda B_j$ (但し, $g_j \in G$)
 - ② $\Sigma_j \tau B_j + \Sigma_k \lambda_k B_k$ (但し, $\lambda_k \in \Sigma$)
 $\rightarrow \Sigma_j h_j^1 \tau B_j + \Sigma_k \lambda_k B_k$ (但し, $h_j^1 \in H^1$)
 - ③ $\Sigma_j \tau B_j$ (但し, $j \geq 2$) $\rightarrow \Sigma_j h_j^1 \tau B_j$ (但し, $h_j^1 \in H^1$)
 - ④ $\Sigma_j \bar{v}_j B_j + \Sigma_k \mu_k B_k$ (但し, $\bar{v}_j \in \bar{A} \cap \Delta, \mu_k \in \Sigma_1$)
 $\rightarrow \Sigma_j h_j^0 \bar{v}_j B_j + \Sigma_k \mu_k B_k$ (但し, $h_j^0 \in H^0$)
 - ⑤ $\Sigma_m v_m B_m + \Sigma_k \mu_k B_k$ (但し, $v_m \in A \cap \Delta$
 かつ $v_m \in L(P_m), \mu_k \in \Sigma_1$)
 $\rightarrow \Sigma_m h_m^0 (\Sigma_j v_m B_j) + \Sigma_k \mu_k B_k$ (但し, $h_m^0 \in H^0$)
- ・プロセス P_i が次の形するとき, 次の置き換えをする。
- ⑥ $P_i = \tau B \rightarrow P_i = h_j^1 \tau B$ (但し, $h_j^1 \in H^1$)
 - ⑦ $P_i = \bar{v} B, P_j = v B$ (但し, $v \in A \cap \Delta$)
 $\rightarrow P_i = h_i^0 \bar{v} B, P_j = v B$ (但し, $h_i^0 \in H^0$)

上記置き換えをしたADEプログラムを $\hat{P} = \hat{P}_1 \parallel \dots \parallel \hat{P}_n$ と表す。

2-2. 半順序集合の生成規則

分岐因子を付与したADEプログラムに対し, 次の二項関係を定義する。

$\xrightarrow{\xi}$: 各 $\xi = (W, \leq)$ に対する (分岐因子付きの) ADEプログラム間の二項関係である。(但し, $W \subseteq \Sigma_2 \cup \{ADE\text{の方の動作識別部} \cup \{NIL\}$,

\leq は W の要素間の順序関係)

また, 便宜的に "END" を ADEプロセスに加える。

以下にその規則を示す。

・各プロセスに対する規則

- (1) NIL $\xrightarrow{\xi_1} \text{END}$, $\xi_1 = (W_1, \leq_1)$ かつ $W_1 = \{NIL\}$
- (2) p $\xrightarrow{\xi_1} \text{END}$, $\xi_1 = (W_1, \leq_1)$ かつ $W_1 = \{q\}$
 (qはpに対応する動作識別ラベル)
 但し, END \rightarrow なるものはない。

- (3) $\hat{P}_1 \xrightarrow{\xi_1} \hat{P}_1$, $\xi_1 = (W_1, \leq_1)$ かつ $W_1 = \phi$

- (4) $\frac{\hat{P}_1 \xrightarrow{\xi_1} \hat{P}_1', \xi_1 = (W_1, \leq_1)}{a \hat{P}_1 \xrightarrow{\xi_2} \hat{P}_1', \xi_2 = (W_2, \leq_2)}$

① $a \in \Sigma_2$ かつ $W_1 \cap \{a\} \times N = \phi$ のとき

$$\textcircled{a} W_2 = W_1 \cup \{a\}$$

$$\textcircled{b} \forall e_1, e_2 \in W_1 \text{ に対して, } e_1 \leq a e_2 \text{ iff } e_1 \leq e_2$$

$$\textcircled{c} \forall e_1 \in W_1 \text{ に対して, } a^1 \leq a e_1$$

② $a \in \Sigma_2$ かつ $W_1 \cap \{a\} \times N = \{a^1, \dots, a^i\}$ のとき

$$\textcircled{a} W_2 = W_1 \cup \{a^{i+1}\}$$

- ⑥ $\forall e_1, e_2 \in W_1 - \{a^1, \dots, a^1\}$ に対して, $e_1 <_{2e} e_2$ iff $e_1 <_1 e_2$
- ⑦ $\forall e_1 \in W_1 - \{a^1, \dots, a^1\}$ に対して, $a^1 <_{2e} a^1$ かつ $(e_1 <_{2a^{j+1}} \text{ iff } e_1 <_1 a^j)$ かつ $(a^{j+1} <_{2e} e_1 \text{ iff } a^j <_1 e_1)$ かつ $a^j <_{2a^{j+1}}$ (但し, $1 \leq j \leq l$)

$$(5) \frac{\hat{p}_1 \xrightarrow{\xi_1} \hat{p}_1'}{\tau \hat{p}_1 \xrightarrow{\xi_2} \hat{p}_1', \xi_1 = \xi_2}$$

$$(6) \frac{\hat{p}_1 \xrightarrow{\xi_1} \hat{p}_1'}{\hat{p}_1 + \hat{p}_2 \xrightarrow{\xi_2} \hat{p}_1', \xi_1 = \xi_2} \cdot \frac{\hat{p}_2 \xrightarrow{\xi_1} \hat{p}_2'}{\hat{p}_1 + \hat{p}_2 \xrightarrow{\xi_2} \hat{p}_2', \xi_1 = \xi_2}$$

・プロセスの合成に対する規則

$$(7) \frac{\hat{p}_1 \xrightarrow{\xi_1} \hat{p}_1', \hat{p}_2 \xrightarrow{\xi_2} \hat{p}_2', \xi_1 = (W_1, <_1), \xi_2 = (W_2, <_2)}{\hat{p}_1 \parallel \hat{p}_2 \xrightarrow{\xi_3} \hat{p}_1' \parallel \hat{p}_2', \xi_3 = (W_3, <_3)}$$

① $W_1 \cap ((L(P_1) \cap \overline{L(P_2)}) \times N) \neq \overline{W_2} \cap ((L(P_1) \cap \overline{L(P_2)}) \times N)$ なら ξ_3 は存在しない。

② $\forall e_1, e_2 \in W_1 \cap ((L(P_1) \cap \overline{L(P_2)}) \times N)$ に対して, $e_1 <_1 e_2$ かつ $\overline{e_2} <_{2e} e_1$ なら ξ_3 は存在しない。

③ $W_3 = W_1 \cup W_2$ 但し, $W_1 = W_1 - W_2 \cap ((L(P_1) \cap \overline{L(P_2)}) \times N)$, $W_2 = W_2 - W_2 \cap ((L(P_1) \cap \overline{L(P_2)}) \times N)$

④ $\forall e_1, e_2 \in W_1', \forall e_3, e_4 \in W_2', \forall x \in W_1 \cap ((L(P_1) \cap \overline{L(P_2)}) \times N)$ に対して, $<_3$ は次のように決まる:

- ① $e_1 <_1 e_2 \rightarrow e_1 <_3 e_2$
- ② $e_3 <_2 e_4 \rightarrow e_3 <_3 e_4$
- ③ $e_1 <_1 x \text{ かつ } \overline{x} <_{2e} e_3 \rightarrow e_1 <_3 e_3$
- ④ $x <_1 e_2 \text{ かつ } e_4 <_{2x} \rightarrow e_4 <_3 e_2$
- ⑤ $h^0 \in H^0 \cap W_1' \text{ かつ } e_4 \in H^0 \text{ かつ } h^0 <_{1x} \text{ かつ } e_4 <_{2x} \rightarrow e_4 <_{3h^0}$
- ⑥ $h^0 \in H^0 \cap W_2' \text{ かつ } e_1 \in H^0 \text{ かつ } e_1 <_{1x} \text{ かつ } h^0 <_{2x} \rightarrow e_1 <_{3h^0}$

$\forall \hat{p}'$ に対して, $\hat{p} \xrightarrow{\xi} \hat{p}'$ なる全ての ξ の集合を Σ と表す。

(P の表現が有限である限り Σ は有限集合である)

2-3. 極大な半順序集合選択規則

ADEP から生成した半順序集合の集合 Σ から次の部分集合を抽出する。

$$\Sigma_m = \{\xi_1, \dots, \xi_m \mid \xi_1 \in \Sigma \text{ and for } \forall \xi \in \Sigma, \xi_1 < \xi\}$$

$<$ は半順序集合間の二項関係で次のように定義される

- $\xi_1 = (W_1, <_1), \xi_2 = (W_2, <_2)$ に対して $\xi_1 < \xi_2$ であるとは, $W_1 \subset W_2$ かつ ① $\forall e_1, e_2 \in W_1$ に対して, $e_1 <_1 e_2$ なら $e_1 <_2 e_2$
- ② $\forall e_1, e_2 \in W_1$ に対して, $e_1 <_1 e_2$ なら $e_1 <_2 e_2$
- ③ $\forall e_1 \in W_1$ に対して, $\forall e_2 \in W_2 - W_1$ なら $e_2 <_{2e} e_1$

但し, 上記 e_2 として NIL は考えない。

$\forall \xi_1 = (W_1, <_1) \in \Sigma_m$ に対して, $W_1 \cap \Sigma \cap \text{last}(\xi_1) \neq \emptyset$ のとき, W_1 を $W_1 \cup \{NIL\}$ に置き換え, $\forall a \in W_1 \cap \Sigma \cap \text{last}(\xi_1)$ に対して, $a <_1 NIL$ の関係を加える。

3. CDE から ADE への変換規則

3-1. トレースの生成規則

{process id.}: プロセスを識別する記号の集合。

(但し, 識別子の中で全順序を付けておく)

process: プロセス割り当てを表す関数で, 任意の Σ_2 の要素 e に対して, $\text{process}(e) = i \in \{\text{process id.}\}$ である。

CDE プログラムを $Q = Q_1 \# \dots \# Q_m$ とする。

各 B インスタンス Q_i に対して次の規則で P^i_k (B インスタンス Q_i に対するプロセス k のトレース) を生成す。

なお, 各 B インスタンス Q_i は半順序集合 $(W(Q_i), <_i)$ に対応させて考える。

- ① $\forall k \in \{\text{process id.}\}$ に対して, $\{e \mid e \in W(Q_i) \text{ and } \text{place}(e) = k\} = \{a^k_1, a^k_2, \dots, a^k_{n_k}\}$ としたとき, $a^k_j <_i a^k_{j+1}$ ($1 \leq j \leq n_k - 1$) なら $P^i_k = (a^k_1 \dots a^k_{n_k})$, $P_{ch}^i_k = \text{emp.}$ 但し, a^k_j は a^k_j の $(\Sigma_2 \cup \Omega_2) \times N \rightarrow \Sigma_2 \cup \Omega_2$ なる射影
- ② $\forall k, m \in \{\text{process id.}\}$ ($k \neq m$) に対して, $a^k_j, a^m_l \in W(Q_i)$ かつ $\text{place}(a^k_j) = k$ かつ $\text{place}(a^m_l) = m$ かつ $a^k_j <_i a^m_l$ かつ $P^i_k = (a^k_1 \dots a^k_{n_k}), P_{ch}^i_k = \text{emp.}(\dots) \dots (\dots)$

$P^i_m = (\dots a^m_1 \dots), P_{ch}^i_m = \text{emp.}(\dots) \dots (\dots)$
 なら $\overline{s < q^k_j, m}, r < q^k_j, l, s < q^k_j, m, r < q^k_j, l, s < q^k_j, m, r < q^k_j, l}$ を Σ の要素に加えて,
 $P^i_k = (\dots a^k_j s < q^k_j, m \dots), P_{ch}^i_k = \text{emp.}(\dots) \dots (\dots)$,
 $P^i_m = (\dots r < q^k_j, l a^m_1 \dots)$,
 $P_{ch}^i_m = \text{emp.}(\dots) \dots (\dots) (s < q^k_j, m, r < q^k_j, l)$

但し, $s < \dots$ は同期信号の送信, $r < \dots$ は同期信号の受信,
 $(s < \dots, r < \dots)$ は信号のキューイングを表す。

$< \dots$ は識別子で q^k_j は次の半順序集合に対してユニークに与えられる。
 $q^k_j = (w_{q_j}, <_q), w_{q_j} = \{e \mid e = a^k_j' \text{ or } (e \in W(Q_i) \text{ and } e <_i a^k_j')\}$,
 for $\forall e_1, e_2 \in w_{q_j}, e_1 <_q e_2$ iff $e_1 <_i e_2$

③ $\forall k \in \{\text{process id.}\}$ に対して,

$$P^i_k = (\dots a^k_j s < q^k_j, m \overline{s < q^k_j, l} \dots) \text{ かつ } 1 \leq m$$

なら $P^i_k = (\dots a^k_j s < q^k_j, l) \overline{s < q^k_j, m} \dots$

また $P^i_k = (\dots r < q^k_j, m, r < q^k_j, l) a^k_j \dots$ かつ $1 \leq m$

なら $P^i_k = (\dots r < q^k_j, l, r < q^k_j, m) a^k_j \dots$

④ 各 P^i_k の $q \in \Omega_q$ は対応する $p \in \Omega_p$ に置き換える。

以上より, 各 i, k ($1 \leq i \leq m, k \in \{\text{process id.}\}$) に対して, $P^i_k, P_{ch}^i_k$ が得られる。

3-2. トレースの合成規則

各 B インスタンスから生成したトレース P^i_k を合成してプロセス P_k を, $P_{ch}^i_k$ からプロセス R_j を導く。

① $\forall k \in \{\text{process id.}\}$ に対して, P^i_k ($1 \leq i \leq m$) から次のように帰納的に $\text{sumform } P_{sum-k}$ を導く。

$$\textcircled{1} i=2 \quad P^i_k = (t_1 \dots t_a u_1 \dots u_b) \text{ で} \\ P^i_k = (t_1 \dots t_a w_1 \dots w_b)$$

(但し, $t_1, u_1, w_1 \in \Sigma_2, u_1 \neq w_1$)

$\rightarrow P_{sum-k} = t_1 \dots t_a (u_1 \dots u_b + w_1 \dots w_b)$ (これは明らかに sumform)

② $i=j-1$ のときの $\text{sumform } P_{sum-k}^{j-1}$ と P^j_k から $i=j$ のときの $\text{sumform } P_{sum-k}^j$ を次のように導く。

S を $\text{sumform}, T$ をトレースとする。

①-1 $S = P_{sum-k}^{j-1}, T = P^j_k$ とすると $S' = P_{sum-k}^j$ である。

①-2 $S = \sum_{i=1, n} a_i B_i, T = (t_1 t_2 \dots t_z)$ とすると,

$$\bullet 1 \leq i \leq n \text{ に対し, } a_i \neq t_i \text{ のとき } S' = \sum_{i=1, n} a_i B_i + T$$

$$\bullet 1 \leq \exists j \leq n \text{ で, } a_j = t_j \text{ のとき } S' = \sum_{i=1, n} a_i B_i + a_j B_j'$$

B_j' は $S = B_j, T = (t_2 \dots t_z)$ として ①-2 を再び繰り返したときの S' に等しい (但し, $z=1$ のとき $B_j' = B_j$)

③ $P_{sum-k} = P_{sum-k}^m$

④ $\forall k \in \{\text{process id.}\}$ の $\text{sumform } P_{sum-k}$ 内の任意の sumform に対して, 次の置き換えをする。

$$\textcircled{1} \sum_{i=1, n} a_i B_i, 1 \leq \exists j \leq n \text{ で } a_j \in G \rightarrow \sum_{i=1, n} a_i B_i + B_j$$

$$\textcircled{2} \sum_{i=1, n} a_i B_i, 1 \leq \exists j \leq n \text{ で } a_j \in H \text{ かつ } B_j = \overline{s < \dots} B_j' \rightarrow \sum_{i=1, n} a_i B_i + B_j$$

$$\textcircled{3} \sum_{i=1, n} a_i B_i, 1 \leq \exists j \leq n \text{ で } B_j = h_j B_j' \text{ かつ } h_j \in H \rightarrow \sum_{i=1, n} a_i B_i + a_j B_j'$$

$$\textcircled{4} \sum_{i=1, n} a_i B_i (n \neq 1), 1 \leq \exists j \leq n \text{ で } a_j \in H \text{ かつ } B_j = \overline{s < \dots} B_j' \rightarrow \sum_{i=1, n} a_i B_i + \tau B_j$$

⑤ ①, ② から導かれた $\forall k \in \{\text{process id.}\}$ の $\text{sumform } P_{sum-k}$ に対して, $P_k = P_{sum-k}$

⑥ $\forall k \in \{\text{process id.}\}$ に対して $P_{ch}^i_k$ ($1 \leq i \leq m$) から次のようにして通信チャンネルに相当するプロセス $R_k^j = s < \dots, r < \dots$ を要素とする集合 RR_k^i ($1 \leq i \leq m$) を生成する。

$$\textcircled{1} P_{ch}^i_k = \text{emp.} (s < \dots, r < \dots) (s < \dots, r < \dots) \dots (s < \dots, r < \dots)$$

とすると, $RR_k^i = \{R_k^{i1}, \dots, R_k^{i2}\}$, 但し, $R_k^{i1} = s < \dots, r < \dots$ ($1 \leq t \leq z$)

$$\textcircled{2} P_{ch}^i_k = \text{emp.} (s < \dots, r < \dots) (s < \dots, r < \dots) \dots (s < \dots, r < \dots)$$

とすると, $RR_k^j = RR_k^{j-1} \cup \{R_k^{j1}, \dots, R_k^{jz}\}$

但し, $R_k^{jt} = s < \dots, r < \dots$ ($1 \leq t \leq y$)

⑦ $\forall k \in \{\text{process id.}\}$ の RR_k^m の和集合をとる。

$$\cup RR_k^m = \{R_1, R_2, \dots, R_z\}$$

(但し, 和集合は $\forall k \in \{\text{process id.}\}$ の全ての k についてとる)

以上により P_k ($k \in \{\text{process id.}\}$) と R_j ($1 \leq j \leq z$) が得られる。
 {process id.} = $\{1, \dots, n\}$ とすると, $ADE P = P_1 \parallel \dots \parallel P_n \parallel R_1 \parallel \dots \parallel R_z$ が導かれる。