ソフトウェア基礎論19－1
プログラミング言語 9－1
（1986. 12. 12）

ＣＣＳsemantics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

ＮＴＴ電気通信研究所

ＮＴＴ　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ＣＣＳsemantics 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，システムの設計段階によって，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，サービス記述はあいまいな表現になったり，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）[8]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，両表現間の変換法と変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（ＡＤＥ＆ＣＤＥ）

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$P_1 \| P_2 \| \cdots \| P_n$（$P_i$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，$Q_1 \# Q_2 \# \cdots \# Q_m$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，これに対なる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．

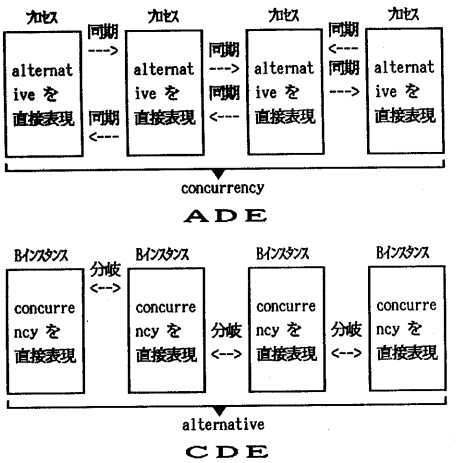


図1．ADE&CDEのイメージ

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，altenative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$Q_1 \# Q_2 \# \cdots \# Q_m$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対応させる．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（外部動作の表示的モデルは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 $P_1 \| P_2 \| \cdots \| P_n$ 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 $Q_1 \# Q_2 \# \cdots \# Q_m$ 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，$\Sigma = \Delta + \overline{\Delta}$，

(For $\forall \lambda \in \Sigma$, $\overline{\lambda} \in \Sigma$ and $((\lambda \in \Delta$ iff $\overline{\lambda} \in \overline{\Delta})$ or $(\lambda \in \overline{\Delta}$ iff $\overline{\lambda} \in \Delta))$ and $\overline{\overline{\lambda}} = \lambda$)

$\Sigma_1 = \Sigma \cup \{\tau\}$ （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 ⇦ E （動作識別ラベルと動作表現式の対応）

E::=NIL｜ e｜λE ｜τE ｜ E+E' ｜E|E' ｜ E＼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,$a \in \Sigma$

$L(E)=s$ $(s \subset \Sigma)$, $L(NIL)=\phi$, $L(aE)=L(E) \cup \{a\}$

$L(e)=L(E)$, $L(\tau E)=L(E)$, $L(E \setminus a)=L(E)-\{a,\overline{a}\}$,

$L(E+E')=L(E) \cup L(E')$, $L(E|E')=L(E) \cup L(E')$,

$L_1(e)=\phi$，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，$names(s)=(s \cup \overline{s}) \cap \Delta$

また，次の省略形を定義する．

$E \setminus a_1 \setminus a_2 \setminus \cdots \setminus a_n = E \setminus A$, $A=\{a_1,a_2,\cdots,a_n\}$

$(E_1|E_2) \setminus A = E_1 \| E_2$ $(A=names(L(E_1) \cap \overline{L(E_2)}))$

【ＡＤＥシンタクス(CCSのｻﾌﾞｾｯﾄ) 】

p：動作識別ラベル （定義した動作識別ラベルの集合をΩ_pとする）

P：動作表現式

$\lambda \in \Sigma$とする．

p ⇦ P （動作識別ラベルと動作表現式の対応）

$P::= P_1 \| P_2 \| \cdots \| P_n$,n:有限整数 (P_i:ﾌﾟﾛｾｽ)

$P_i::=$ NIL｜ p｜λP_i｜τP_i｜ 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$L(p) \cap \overline{L(P_j)}=\phi$ $(i \neq j)$

(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．

$1 \leqq \forall i,j,k \leqq n$ に対して

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

【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" ことを表す．

(1) $\mu P_i \xrightarrow{\mu} P_i$

(2) $$\frac{P_1 \xrightarrow{\mu} P_1'}{P_1+P_2 \xrightarrow{\mu} P_1'} \qquad \frac{P_2 \xrightarrow{\mu} P_2'}{P_1+P_2 \xrightarrow{\mu} P_2'}$$

for $1 \leqq \forall i \leqq n, \forall \mu \notin A$ $(A=name(\cup_{j \neq k}(L(P_j) \cap \overline{L(P_k)}))$,

(3) $$\frac{P_i \xrightarrow{\mu} P_i'}{P_1 \| P_2 \| \cdots \| P_n \xrightarrow{\mu} P_1 \| P_2 \| \cdots \| P_i' \| \cdots \| P_n}$$

(4) $$\frac{\text{for } 1 \leqq \forall i \leqq n,\ P_i \xrightarrow{\lambda} P_i' \text{ and } P_j \xrightarrow{\overline{\lambda}} P_j'}{P_1 \| \cdots \| P_n \xrightarrow{\tau} P_1 \| \cdots \| P_i' \| \cdots \| P_j' \| \cdots \| P_n}$$

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

(6)

3.2 ＣＤＥ

ＣＤＥは動作のalternative を$Q_1 \# Q_2 \# \cdots \# Q_m$のようにＢインスタンスの組で表す．そこで，図２ⓐの動作はＣＤＥではⓐ' のように表す．ⓐ' がⓑ，ⓒではなくⓐ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．ⓑ，ⓒ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（ⓑのような動作に対しては $g_1,g_2 \in G$を付与してⓑ' のように，ⓒのような動作に対しては $h_1 \in H$を付与してⓒ' 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．

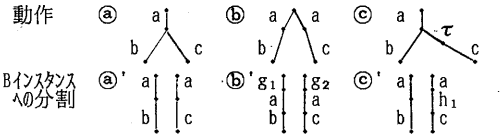


図２．非決定的動作の例

【領域】

$\Sigma_2=\Sigma \cup G \cup H$ （Ｎは自然数，Σ，Ｇ，Ｈは共通部分を持たない）

$\Sigma_3=\Sigma_2 \times N$

簡単のため任意の要素(a,i) $\in \Sigma_3$ をa^iと表す．

また，$\overline{a^i}$は$(\overline{a},i)$ 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の集合をΩ_qとする）

Q：動作表現式

"#", "≪", "," はＣＤＥのオペレータ

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

q ⇦ Q （動作識別ラベルと動作表現式の対応）

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

$Q_i::= \{ R \}$

R::= NIL ｜q^j｜ $\alpha \ll$NIL ｜ $\alpha \ll q^j$｜q^j,R｜

$\alpha \ll$NIL,R ｜ $\alpha \ll q^j$,R ｜ $\alpha \ll \alpha'$,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$\ll_i$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

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

$\ll_i$: $W(Q_i)$の要素間の二項関係で次のように定義する

Q_iが{ …, $a_1 \ll a_m$, … } なら　$a_1 \ll_i a_m$

$a_1 \ll_i a_m$ かつ　$a_m \ll_i a_n$ なら　$a_1 \ll_i a_n$

各Q_iは次の条件を満足しなくてはならない．

(a)　$\forall a_1, a_m \in W(Q_i)$ で　$a_1 \ll_i a_m$ なら　$a_m \not\ll_i a_1$

(b)　$\forall a_1 \in W(Q_i)$ で，　$a_1 \not\ll_i a_1$

(c)　$\forall a_1 \in W(Q_i) \cap \Sigma_3$ で，$a_1 \ll_i NIL$か　$a_1 \ll_i q^j$ □

このときQ_iに対して半順序集合 $(W(Q_i), \ll_i)$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 $(W(Q_i), \ll_i)$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$\xi = (W, \ll)$ に対して次の定義をする．

first: W の最も順序の小さい要素を抽出する関数．

$$first(\xi) = \{a_1 \in W \mid \text{for } \forall a_m \in W \ a_m \not\ll_i a_1\}$$

last : W の最も順序の大きい要素を抽出する関数．

$$last(\xi) = \{a_1 \in W \mid \text{for } \forall a_m \in W \ a_1 \not\ll_i a_m\}$$

$\ll^d$: $a \ll^d b$　とは，　$a \ll b$ で，　$a \ll c \ll b$ 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)
$$\frac{\alpha \in first(Q_i) \cap \Sigma_3}{Q_i \xrightarrow{\alpha}> Q_i',\ W(Q_i') = W(Q_i) - \{\alpha\} \text{ and } (\text{for } \forall a_j, a_k \in W(Q_i'),\ a_j \ll_i' a_k \text{ iff } a_j \ll_i a_k \ (\ll_i \text{はQ}_i\text{の}, \ll_i' \text{はQ}_i' \text{の順序関係}))}$$

(3)
$$\frac{\text{for } \forall a^i \in \Sigma \times N,\ \exists J \subseteq \{1, \cdots, m\} (J = \{j1, \cdots, jn\},\ \underline{J} = \{1, \cdots, m\} - J) \quad \text{for } \forall j \in J,\ Q_j \xrightarrow{a^i}> Q_j' \text{ and } \text{for } \forall k \in \underline{J},\ Q_k - x \xrightarrow{a^i}> Q_k'}{Q_1 \# Q_2 \# \cdots \# Q_m \xrightarrow{a}> Q_{j1}' \# \cdots \# Q_{jn}'}$$

(4)
$$\frac{\text{for } \forall g^p \in G \times N,\ \forall a^i \in \Sigma \times N,\ \exists J \subseteq \{1, \cdots, m\} (J = \{j1, \cdots, jn\},\ \underline{J} = \{1, \cdots, m\} - J) \quad \text{for } \forall j \in J,\ Q_j \xrightarrow{g^p}> Q_j' \xrightarrow{a^i}> Q_j'' \text{ and } g^p \ll_j a^i \quad \text{and} \quad \text{for } \forall k \in \underline{J} \quad Q_k - x \xrightarrow{g^p}> Q_k'}{Q_1 \# Q_2 \# \cdots \# Q_m \xrightarrow{a}> Q_{j1}'' \# \cdots \# Q_{jn}''}$$

(5)
$$\frac{\text{for } \forall h^p \in H \times N,\ \exists J \subseteq \{1, \cdots, m\} (J = \{j1, \cdots, jn\},\ \underline{J} = \{1, \cdots, m\} - J) \quad \text{for } \forall j \in J,\ Q_j \xrightarrow{h^p}> Q_j' \text{ and } \text{for } \forall k \in \underline{J},\ Q_k - x \xrightarrow{h^p}> Q_k'}{Q_1 \# Q_2 \# \cdots \# Q_m \xrightarrow{\tau}> Q_{j1}' \# \cdots \# Q_{jn}'}$$

(6)
$$\frac{\text{for } \forall q^i \in \Omega_q \times N,\ \exists J \subseteq \{1, \cdots, m\} (J = \{j1, \cdots, jn\},\ \underline{J} = \{1, \cdots, m\} - J) \quad \text{for } \forall j \in J,\ q^i \in first(Q_j) \text{ and } \text{for } \forall k \in \underline{J},\ q^i \notin first(Q_k) \quad \text{and } q \Leftarrow Q \text{ and } Q \xrightarrow{a}> Q' \quad (Q' = Q_1' \# \cdots \# Q_s',\ a \in \Sigma)}{\text{for } \forall j \in J,\ Q_j \xrightarrow{a^{n+i}}> Q'', Q'' = Q_1'' \# \cdots \# Q_s'',\ \text{for } 1 \leqq \forall r \leqq s,\ W(Q_r'') = (W(Q_j) - \{q^i\}) \cup W'(Q_r') \ (\text{for } \forall b^p \in (\Sigma_2 \cup \Omega_q) \times N,\ b^{p+n} \in W'(Q_r') \text{ iff } b^p \in W(Q_r')) \text{ and } (\text{for } \forall \alpha, \beta \in (W(Q_j) - \{q^i\}),\ \alpha \ll_r'' \beta \text{ iff } \alpha \ll_j \beta \text{ and for } \forall b^v, c^w \in W(Q_r'),\ b^{v+n} \ll_r'' c^{w+n} \text{ iff } b^v \ll_r' c^w \ (\ll_j \text{はQ}_j\text{の}, \ll_r' \text{はQ}_r' \text{の}, \ll_r'' \text{はQ}_r'' \text{の順序関係})) \quad \text{但し，} n = \max(\{x \mid \text{for } \forall j \in J,\ \forall b^x \in W(Q_j) \text{のNへの射影}\})}$$

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[8]である

(2) 変換の前後で並列性は保存される（付録参照）□

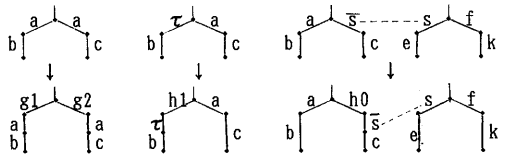
なお，本章ではＡＤＥ，ＣＤＥの各プロセス，Ｂインスタンスを簡単のために，それぞれ tree(アークにイベントを対応させる)，Hasse図形（ノードにイベントを対応させ，順序は下方向で矢印は略す）で表す．

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】

①ＡＤＥ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）

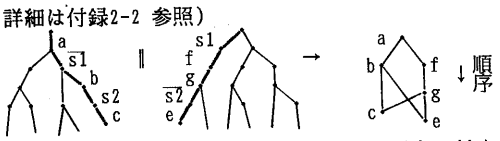


但し，　$g1, g2 \in G$，$h1 \in H^1$，　$h0 \in H^0$

・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　$H^0 \cap H^1 = \phi$ 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．

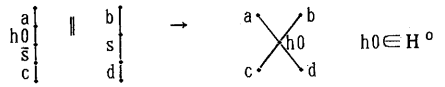
②分岐因子を付与した各プロセスの分岐の選択子ごとに順序関係を調べて，半順序集合を生成する．（詳細は付録2-2 参照）



・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．(τ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 $(a\bar{s}) \,||\, (sb) \rightarrow a \ll b$)

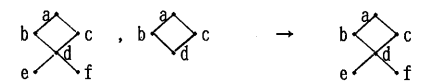
・H^0 の要素は次のような順序関係に変換する．



・names$(L(P_i) \cap \overline{L(P_j)})$の要素で，同期相手のないイベントが現れた時点（デットロック）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,

$\forall a \in (\Sigma_3 \times N) \cap last(\xi)$ に対して, $a \ll NIL$ 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_iに置き換えられ,

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

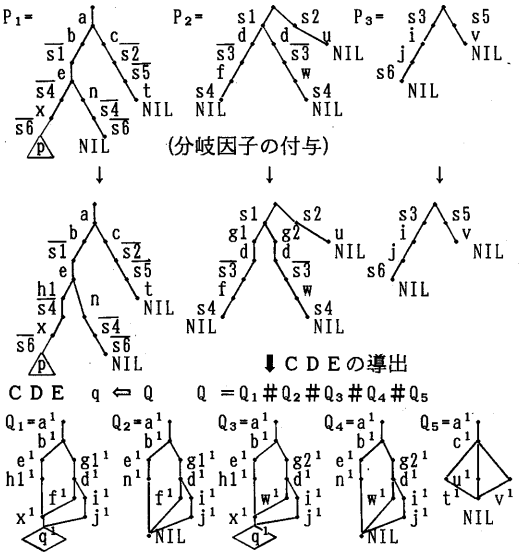
【変換の正当性】

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　$p \Leftarrow P$　　$P = P_1 \| P_2 \| P_3$



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.

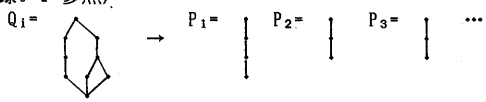
②プロセス数最小等, 所要リソースや実行時間に対する最適化条件を満たす変換方法を求める.

②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条件を満足するＡＤＥを導けるとは限らない. プロセスに振り分けたとき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に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.

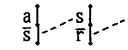
【変換規則】

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, $\Omega_q \times N$ 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,

$a \ll^d b \rightarrow$ 

のようにする. ($\bar{s}$,s,$\bar{r}$,r をΣに加える)

( では cと dの間にも順序関係がつく)

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及び, 異なるプロセスに属するイベント間の順序関係のために生成した sと rから成るトレースは, 容易に3.1 のシンタクスを満たすプロセスP_iに置き換えられ, ＡＤＥプログラム $P = P_1 \# \cdots \# P_n$が得られる.

【変換条件】

ＣＤＥプログラムを $Q = Q_1 \# \cdots \# Q_m$　とする.

[条件ⓐ] 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, $process(e_1) = process(e_2)$ なら　　$e_1 \ll_i e_2$ か　$e_2 \ll_i e_1$ である.

[条件ⓑ]　$\{Q_1, Q_2, \cdots, Q_m\}$は "分配可能" である.

$\{Q_1, Q_2, \cdots, Q_m\}$が分配可能とは,

$\{Q_1, Q_2, \cdots, Q_m\}$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.

$\{Q_1, Q_2, \cdots, Q_m\} \longrightarrow \underset{G1}{\{Q^1_1, \cdots, Q^1_{m1}\}}, \cdots, \underset{Gk}{\{Q^k_1, \cdots, Q^k_{mk}\}}$

(G1, …,Gk はグループ名).

条件 : $\forall Gi$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)　$\forall Q_j, Q_k \in Gi$に対し, $a_i \in W(Q_j) \cap W(Q_k)$

(2)　$\forall Gi, Gj$ $(i \neq j)$に対して, $a_i \neq a_j$

(3)　$\forall Q_x \in Gi, \forall Q_y \in Gj$ $(i=j$ も可) に対して

$(e_1,e_2 \in W(Q_x)$ で $e_1,e_2 \lessdot_x a_i$ かつ $e_1 \lessdot_x e_2)$
iff $(e_1,e_2 \in W(Q_y)$ で $e_1,e_2 \lessdot_y a_j$ かつ $e_1 \lessdot_y e_2)$

(4) $\forall Q_x \in Gi$対して，$\exists Q_y \in Gj$，
$(e_1,e_2 \in W(Q_x)$ で，$a_i \not\lessdot_x e_1,e_2$ かつ $e_1 \lessdot_x e_2)$
iff $(e_1,e_2 \in W(Q_y)$ で，$a_j \not\lessdot_y e_1,e_2$ かつ $e_1 \lessdot_y e_2)$
（但し，$e_1,e_2 \notin \{a_i, a_j\}$）

[条件ⓒ] 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．

$\{Q_1,Q_2,\cdots,Q_m\} \longrightarrow \underbrace{\{Q^1_1,\cdots,Q^1_{m1}\}}_{G1},\cdots,\underbrace{\{Q^k_1,\cdots,Q^k_{mk}\}}_{Gk}$

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

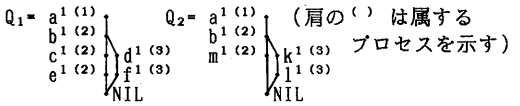
各条件は次のことを意味する．
ⓐ同一プロセスに属するΣ_3 の要素間には必ず順序がつく．
ⓑＣＤＥのＢインスタンス間には動作が分岐する部分に相当するΣ_3 の要素（分配点）の対応関係が存在して，一つの対応関係（分岐）は他の対応関係（分岐）に対して因果関係があるか完全に独立かである．
ⓒ対応する分配点は同一プロセスに属する．
ⓐ，ⓒはプロセス割り当てに対する条件である．ⓑ，ⓒは十分条件である．ⓑ，ⓒを満たさなくてもＡＤＥに変換可能なものもある．しかし，このようなものは，「alternative はプロセス内に表現される」というＡＤＥの基本思想からはずれるものである．特にⓑ，ⓒの条件を以降，"完全性の条件" 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】

Q＝Q_1＃Q_2

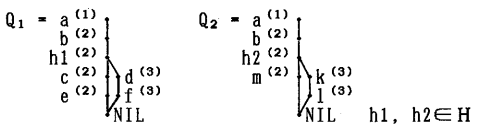


（肩の( ) は属するプロセスを示す）

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）．

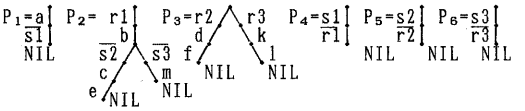
[case 1] 一つのプロセスにおいて観測可能でない要素で分岐する．

Q＝Q_1＃Q_2



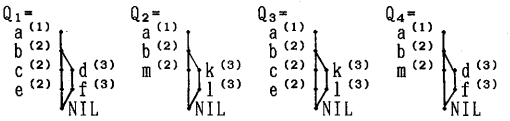
分配点は h1 と h2．⬇ＡＤＥの導出

P= P_1＃P_2＃P_3＃P_4＃P_5＃P_6



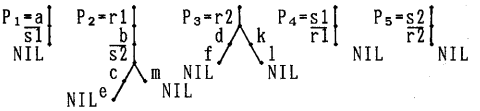
[case 2]プロセス２とプロセス１で独立に分岐する．

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，変換に際して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-XVIII
(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 → (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，"$\xrightarrow{s}$>"，"≈"，"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 「$P=P_1\|\cdots\|P_n$ が$\lambda_1\in L(P_i),\lambda_2\in L(P_j)$ で，
$P_1\|\cdots\|P_n\xrightarrow{s}\cdots\|P_i'\|\cdots\|P_j'\|\cdots$　かつ
$\cdots\|P_i'\|\cdots\|P_j'\|\cdots\xrightarrow{\lambda_1}\cdots\|P_i''\|\cdots$ かつ
$\cdots\|P_i'\|\cdots\|P_j'\|\cdots\xrightarrow{\lambda_2}\cdots\|P_j''\|\cdots$」（状態１）のとき
「$Q=Q_1\#\cdots\#Q_m$ に対して，$Q_{k1}'\#\cdots\#Q_{ks}'$が存在し，
$Q_1\#\cdots\#Q_m\xrightarrow{s}Q_{k1}'\#\cdots\#Q_{ks}'$　かつ
その中に$\lambda_1,\lambda_2\in$first(Q_{ki}') なるインスタンスQ_{ki}'が存在する（即ち，Q_{ki}'を半順序集合 $(W(Q_{ki}'),\prec_{ki})$ 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　$\cdots\|P_i'\|\cdots\|P_j'\|\cdots\approx Q_{k1}'\#\cdots\#Q_{ks}'$　である．

②逆に，$Q=Q_1\#\cdots\#Q_m$に上記のような $Q_{k1}'\#\cdots\#Q_{ks}'$（状態２）があるとき，$P=P_1\|\cdots\|P_n$に対して，上記のような$\cdots\|P_i'\|\cdots\|P_j'\|\cdots$（状態１）が存在して，$\cdots\|P_i'\|\cdots\|P_j'\|\cdots\approx Q_{k1}'\#\cdots\#Q_{ks}'$　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$B_1+\cdots+B_n$ なる表現をΣ_iB_i と略す．
［定義］（sumform） B がsumform であるとは，$\Sigma_ig_iB_i$ 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p はsumform である）　□
$H=H^0\cup H^1$ かつ　$H^0\cap H^1=\phi$
［分岐因子付与規則］
ADEプログラムを$P=P_1\|\cdots\|P_n=(P_1|\cdots|P_n)\setminus A$ 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 $j\geqq 2$）　→　$\Sigma_j g_j\lambda B_j$（但し，$g_j\in G$）
②　$\Sigma_j\tau B_j+\Sigma_k\lambda_kB_k$　（但し，$\lambda_k\in\Sigma$）
→　$\Sigma_j h_j{}^1\tau B_j+\Sigma_j\lambda_kB_k$　（但し，$h_j{}^1\in H^1$）
③　$\Sigma_j\tau B_j$（但し，　$j\geqq 2$）　→　$\Sigma_jh_j{}^1\tau B_j$　（但し，$h_j{}^1\in H^1$）
④　$\Sigma_j\overline{\nu}_jB_j+\Sigma_k\mu_kB_k$　（但し，$\overline{\nu}_j\in\overline{A}\cap\overline{\Delta}$，$\mu_k\in\Sigma_1$）
→　$\Sigma_jh_j{}^o\overline{\nu}_jB_j+\Sigma_k\mu_kB_k$（但し，$h_j{}^o\in H^o$）
⑤　$\Sigma_{m\,j}\nu_{mj}B_{mj}+\Sigma_k\mu_kB_k$　（但し，$\nu_{mj}\in A\cap\Delta$
かつ　$\nu_{mj}\in L(P_m)$，$\mu_k\in\Sigma_1$）
→　$\Sigma_mh_m{}^o(\Sigma_j\nu_{mj}B_j)+\Sigma_k\mu_kB_k$（但し，$h_m{}^o\in H^o$）
・プロセスP_iが次の形のとき，次の置き換えをする．
⑥　$P_i=\tau B$　→　$P_i=h_j{}^1\tau B$　（但し，$h_j{}^1\in H^1$）
⑦　$P_i=\overline{\nu}B_i$，$P_j=\nu B_j$　（但し，$\nu\in A\cap\Delta$）
→　$P_i=h_i{}^o\overline{\nu}B_i$，　$P_j=\nu B_j$　（但し，$h_i{}^o\in H^o$）

上記置き換えをしたADEプログラムを　$\hat{P}=\hat{P}_1\|\cdots\|\hat{P}_n$　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$\xi=(W,\prec)$ 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，$W\subset\Sigma_2\cup\{$ADEプログラムの動作識別$\}\cup\{$NIL$\}$，
$\prec$ 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 $\xrightarrow{\xi_1}$END，　$\xi_1=(W_1,\prec_1)$　かつ $W_1=\{$NIL$\}$
(2) p $\xrightarrow{\xi_1}$END，　$\xi_1=(W_1,\prec_1)$　かつ　$W_1=\{q\}$
（qはp に対応する動作識別ラベル）
但し，　END---->　なるものはない．
(3)　$\hat{P}_i\xrightarrow{\xi_1}\hat{P}_i$，　$\xi_1=(W_1,\prec_1)$　かつ　$W_1=\phi$
(4)　$\dfrac{\hat{P}_i\xrightarrow{\xi_1}\hat{P}_i',\ \xi_1=(W_1,\prec_1)}{a\hat{P}_i\xrightarrow{\xi_2}\hat{P}_i',\ \xi_2=(W_2,\prec_2)}$
① $a\in\Sigma_2$　かつ　$W_1\cap\{a\}\times N=\phi$のとき
ⓐ $W_2=W_1\cup\{a^1\}$
ⓑ $\forall e_1,e_2\in W_1$に対して，$e_1\prec_2e_2$ iff $e_1\prec_1e_2$
ⓒ $\forall e_1\in W_1$に対して $,a^1\prec_2e_1$
② $a\in\Sigma_2$　かつ　$W_1\cap\{a\}\times N=\{a^1,\cdots,a^i\}$のとき
ⓐ $W_2=W_1\cup\{a^{i+1}\}$

ⓑ $\forall e_1,e_2 \in W_i-\{a^1,\cdots,a^j\}$対して, $e_1 \lessdot_z e_2$ iff $e_1 \lessdot_i e_2$

ⓒ $\forall e_1 \in W_i-\{a^1,\cdots,a^j\}$に対して ,$a^j \lessdot_z e_1$ かつ

$(e_1 \lessdot_z a^{j+1}$ iff $e_1 \lessdot_i a^j)$ かつ $(a^{j+1} \lessdot_z e_1$ iff $a^j \lessdot_i e_1)$ かつ

$a^j \lessdot_z a^{j+1}$ (但し, $1 \leqq j \leqq i$)

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

(6) $$\frac{\hat{P}_i \xrightarrow{\xi_1} \hat{P}_i'}{\hat{P}_i+\hat{P}_j \xrightarrow{\xi_2} \hat{P}_i' \ ,\xi_1=\xi_2} \ , \quad \frac{\hat{P}_j \xrightarrow{\xi_1} \hat{P}_j'}{\hat{P}_i+\hat{P}_j \xrightarrow{\xi_2} \hat{P}_j' \ ,\xi_1=\xi_2}$$

・プロセスの合成に対する規則

(7) $$\frac{\hat{P}_i \xrightarrow{\xi_1} \hat{P}_i' \quad , \quad \hat{P}_j \xrightarrow{\xi_2} \hat{P}_j' \ ,\xi_1=(W_1,\lessdot_1),\ \xi_2=(W_2,\lessdot_2)}{\hat{P}_i \| \hat{P}_j \xrightarrow{\xi_3} \hat{P}_i' \| \hat{P}_j' \quad ,\xi_3=(W_3,\lessdot_3)}$$

①$W_1 \cap (\ (L(P_i) \cap \overline{L(P_j)}) \times N) \neq \overline{W_2} \cap (\ (L(P_i) \cap \overline{L(P_j)}) \times N)$なら

ξ_3 は存在しない.

② $\forall e_1,e_2 \in W_1 \cap (\ (L(P_i) \cap \overline{L(P_j)}) \times N)$に対して,

$e_1 \lessdot_1 e_2$ かつ $\overline{e_2} \lessdot_2 \overline{e_1}$ なら ξ_3 は存在しない.

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

$W_2'=W_2 - W_2 \cap (\ (\overline{L(P_i)} \cap L(P_j)) \times N)$

④ $\forall e_1,e_2 \in W_1'$, $\forall e_3,e_4 \in W_2'$, $\forall x \in W_1 \cap (\ (L(P_i) \cap \overline{L(P_j)}) \times N)$に対して, $\lessdot_3$ は次のように決まる:

ⓐ $e_1 \lessdot_1 e_2 \ \rightarrow \ e_1 \lessdot_3 e_2$

ⓑ $e_3 \lessdot_2 e_4 \ \rightarrow \ e_3 \lessdot_3 e_4$

ⓒ $e_1 \lessdot_1 x$, かつ $\overline{x} \lessdot_2 e_3 \ \rightarrow \ e_1 \lessdot_3 e_3$

ⓓ $x \lessdot_1 e_2$ かつ $e_4 \lessdot_2 \overline{x} \ \rightarrow \ e_4 \lessdot_3 e_2$

ⓔ $h^0 \in H^0 \cap W_1'$ かつ $e_4 \notin H^0$ かつ $h^0 \lessdot^d_1 x$ かつ $e_4 \lessdot^d_2 \overline{x}$

$\rightarrow \ e_4 \lessdot_3 h^0$

ⓕ $h^0 \in H^0 \cap W_2'$ かつ $e_1 \notin H^0$ かつ $e_1 \lessdot^d_1 x$ かつ $h^0 \lessdot^d_2 \overline{x}$

$\rightarrow \ e_1 \lessdot_3 h^0$

$\forall \hat{P}'$に対して, $\hat{P} \xrightarrow{\xi} \hat{P}'$ なる全てのξの集合をΞと表す.

(P の表現が有限である限りΞは有限集合である)

<u>2-3.極大な半順序集合選択規則</u>

ADEP から生成した半順序集合の集合Ξから次の部分集合を抽出する.

$\Xi_m=\{\xi_1,\cdots,\xi_m \mid \xi_i \in \Xi$ and for $\forall \xi \in \Xi,\ \xi_i \not< \xi\}$

$<$は半順序集合間の二項関係で次のように定義される

$\xi_1=(W_1,\lessdot_1),\ \xi_2=(W_2,\lessdot_2)$に対して $\xi_1 < \xi_2$ であるとは,

$W_1 \subseteq W_2$かつⓐ $\forall e_1,e_2 \in W_1$に対して, $e_1 \lessdot_1 e_2$ なら $e_1 \lessdot_2 e_2$

ⓑ $\forall e_1,e_2 \in W_1$に対して, $e_1 \lessdot_1 e_2$ なら $e_1 \lessdot_2 e_2$

ⓒ $\forall e_1 \in W_1$に対して, $\forall e_3 \in W_2 - W_1$ なら $e_3 \lessdot_2 e_1$

但し, 上記e_iとしてNIL は考えない.

$\forall \xi_i=(W_i, \lessdot_i) \in \Xi_m$ に対して, $W_i \cap \Sigma_3 \cap$ last$(\xi_i) \neq \phi$のとき,

W_iを$W_i \cup \{NIL\}$ に置き換え,

$\forall a \in W_i \cap \Sigma_3 \cap$ last(ξ_i)に対して, $a \lessdot_i$NILの関係を加える.

3. CDEからADEへの変換規則

<u>3-1.トレースの生成規則</u>

{process id.}:プロセスを識別する記号の集合.

(但し, 識別子の間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,

process(e)=i$\in$ {process id.} である.

CDEプログラムを $Q=Q_1 \# \cdots \# Q_m$ 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$(W(Q_i), \lessdot_i)$に対応させて考える.

① $\forall$ k$\in$ {process id.} に対して,

{e| e$\in W(Q_i)$ and place(e)=k}$=\{a^k_1', a^k_2', \cdots, a^k_{nk}'\}$ としたとき,

$a^k_j' \lessdot_i a^k_{j+1}'$ $(1 \leqq j \leqq nk-1)$ なら $P^i_k=(a^k_1 \cdots a^k_{nk})$, $P_{ch}{}^i{}_k$=emp.

但し, a^k_j は $a^k_j{}'$ の $(\Sigma_2 \cup \Omega_q) \times N \rightarrow \Sigma_2 \cup \Omega_q$ なる射影

② $\forall$ k,m$\in$ {process id.}$(k \neq m)$に対して,

$a^k_j{}', a^m_1{}' \in W(Q_i)$ かつ place$(a^k_j{}')$=k かつ place$(a^m_1{}')$=mで

$a^k_j{}' \lessdot_i {}^d a^m_1{}'$かつ

$P^i_k=(\cdots a^k_j \cdots)$, $P_{ch}{}^i{}_k$=emp.$(\cdots) \cdots (\cdots)$,

$P^i_m=(\cdots a^m_1 \cdots)$, $P_{ch}{}^i{}_m$=emp.$(\cdots) \cdots (\cdots)$

なら $s<q^k_j,m>$,$r<q^k_j,k>$,$s<q^k_j,m>$,$\overline{r<q^k_j,k>}$ をΣの要素に加えて,

$P^i_k=(\cdots a^k_j \ \overline{s<q^k_j,m>} \cdots)$, $P_{ch}{}^i{}_k$=emp.$(\cdots) \cdots (\cdots)$,

$P^i_m=(\cdots r<q^k_j,k> \ a^m_1 \cdots)$,

$P_{ch}{}^i{}_m$=emp.$(\cdots) \cdots (\cdots)(s<q^k_j,m> \ \overline{r<q^k_j,k>})$

但し, $\overline{s<\cdots>}$ は同期信号の送信, $r<\cdots>$ は同期信号の受信,

$(s<\cdots> \ \overline{r<\cdots>})$は信号のキューイングを表す.

$<\cdots>$ 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. $q^k_j=(W_q, \lessdot_q)$, $W_q=\{e \mid e=a^k_j{}'$ or$(e \in W(Q_i)$and $e \lessdot_i a^k_j{}')\}$,

for $\forall e_1,e_2 \in W_q$, $e_1 \lessdot_q e_2$ iff $e_1 \lessdot_i e_2$

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

$P^i_k=(\cdots a^k_j \ \overline{s<q^k_j,m>} \ \overline{s<q^k_j,l>} \cdots)$ かつ $l<m$

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

また $P^i_k=(\cdots r<q^m_u,m> \ r<q^l_v,l> \ a^k_j \cdots)$ かつ $l<m$

なら $P^i_k=(\cdots r<q^l_v,l> \ r<q^m_u,m> \ a^k_j \cdots)$

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 $(1 \leqq i \leqq m,\ k \in$ {process id.}$)$に対して, P^i_k ,$P_{ch}{}^i{}_k$ が得られる.

<u>3-2.トレースの合成規則</u>

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

ⓐ i=2 $P^1_k=(t_1 \cdots t_a \ u_1 \cdots u_b)$ で

$P^2_k=(t_1 \cdots t_a \ w_1 \cdots w_c)$

(但し, $t_i,u_i,w_i \in \Sigma_2$, $u_i \neq w_i$)

$\rightarrow P_{sum-k}{}^2=t_1 \cdots t_a(u_1 \cdots u_b + w_1 \cdots w_c)$ (これは明らかに sumform)

ⓑ i=j-1のときの sumform $P_{sum-k}{}^{j-1}$と P^j_k から

i=j のときの sumform $P_{sum-k}{}^j$を次のように導く.

Sを sumform, T をトレースとする.

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

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

- $1 \leqq \forall i \leqq nn$ に対し, $a_i \neq t_1$のとき $S'=\sum_{i=1,\ nn} a_i B_i+T$
- $1 \leqq \exists j \leqq nn$ で, $a_j=t_1$のとき $S'=\sum_{i \neq j} a_i B_i+a_j B_j'$

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

ⓒ $P_{sum-k}=P_{sum-k}{}^m$

② $\forall k \in$ {process id.} の sumform P_{sum-k} 内の任意の sumformに対して, 次の置き換えをする.

ⓐ $\sum_{i=1,\ n} a_i B_i$, $1 \leqq \exists j \leqq n$ で $a_j \in G \ \rightarrow \ \Sigma a_i B_i+B_j$

ⓑ $\sum_{i=1,\ n} a_i B_i$, $1 \leqq \exists j \leqq n$ で $a_j \in H$ かつ $B_j=s<\cdots>B_j' \rightarrow \sum_{i \neq j} a_i B_i+B_j$

ⓒ $\sum_{i=1,\ n} a_i B_i$, $1 \leqq \exists j \leqq n$ で $B_j=h_j B_j'$ かつ $h_j \in H \rightarrow \sum_{i \neq j} a_i B_i+a_j B_j'$

ⓓ $\sum_{i=1,\ n} a_i B_i (n \neq 1)$, $1 \leqq \exists j \leqq n$ で $a_j \in H$ かつ $B_j \neq s<\cdots>B_j'$

$\rightarrow \sum_{i \neq j} a_i B_i+\tau B_j$

③ ①, ②から導かれた $\forall k \in$ {process id.} のsumform P_{sum-k}に対して,

$P_k=P_{sum-k}$

④ $\forall k \in$ {process id.} に対して$P_{ch}{}^i{}_k$ $(1 \leqq i \leqq m)$から次のようにして通信チャネルに相当するプロセス$R_k{}^j=s<\cdots> \ \overline{r<\cdots>}$ を要素とする集合$RR_k{}^i$ $(1 \leqq i \leqq m)$を生成する.

ⓐ $P_{ch}{}^1{}_k$=emp. $(s<\cdots> \overset{A1}{\overline{r<\cdots>}})(\ s<\cdots> \overset{A2}{\overline{r<\cdots>}}\ ) \cdots (\ s<\cdots> \overset{Az}{\overline{r<\cdots>}}\ )$

とすると, $RR_k{}^1=\{R_k{}^{A1}, \cdots, R_k{}^{Az}\}$, 但し, $R_k{}^{At}=s<\cdots> \overset{At}{\overline{r<\cdots>}}$ $(1 \leqq t \leqq z)$

ⓑ $P_{ch}{}^j{}_k$=emp. $(s<\cdots> \overset{B1}{\overline{r<\cdots>}})(\ s<\cdots> \overset{B2}{\overline{r<\cdots>}}\ ) \cdots (\ s<\cdots> \overset{By}{\overline{r<\cdots>}}\ )$

とすると, $RR_k{}^j=RR_k{}^{j-1} \cup \{R_k{}^{B1}, \cdots, R_k{}^{By}\}$,

但し, $R_k{}^{Bt}=s<\cdots> \overset{Bt}{\overline{r<\cdots>}}$ $(1 \leqq t \leqq y)$

⑤ $\forall k \in$ {process id.} の kに対して$RR_k{}^m$の和集合をとる.

$\bigcup_k RR_k{}^m=\{R_1, R_2, \cdots, R_w\}$ とする.

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

以上により P_k (k$\in$ {process id.}) と R_j $(1 \leqq j \leqq z)$が得られる.

{process id.}$=\{1, \cdots, n\}$ とすると, ADE $P=P_1 \| \cdots \| P_n \| R_1 \| \cdots \| R_z$ が導かれる.