ソフトウェア基礎論　25－9
（1988. 5. 26）

通信の動的な名前づけによるＣＣＳの拡張

Extending CCS with Dynamaic Communication Naming

結縁祥治　　坂部俊樹　　稲垣康善
Shoji YUEN　　Toshiki SAKABE　　Yasuyoshi INAGAKI

名古屋大学　工学部
Faculty of Engineering, Nagoya University

yuen@alsun50.inagaki.nagoya-u.junet

あらまし　本稿では，Milnerの提案したＣＣＳにおいて，通信の名前づけを実行時に動的に決定できるように拡張を行う．このような動的な名前づけは，ＣＣＳによる実際的な並行処理システムのモデル化においては自然な拡張である．そのようなモデルでは，動作体間で通信が起こったときに受け渡される値によって，それ以降に起こる通信の名前が決定する．本稿における拡張では，動的に変化する通信の名前を変数を含むような項によって表現する．拡張された動作式の意味は，名前を表現する項から実際の名前の領域への写像を，項のトップレベルの関数記号に割り当てることによって定義される．拡張された動作式の意味論はMilnerのＣＣＳに基づいて与えられる．最後に，このような名前の割り当ておよび拡張動作式における制限演算について議論する．

Abstract　In this paper, we extend Milner's CCS with dynamic communication naming, which is a natural extension in view of modelling practical concurrent systems by CCS. The communication names are to be specified by passing values between agents. We propose to represent the communication names as terms allowed to have variables. The semantics of this extended CCS is given based on the original CCS by assigning to the top-level function symbols the mappings from ground terms to the name-domain values. Finally, we discuss this name assignment and the restriction operation for dynamic communication naming.

1.　はじめに

Milnerの提案したＣＣＳ(A Calculus of Communicating Systems)[1]は並行計算を抽象化した計算体系であり，並行計算に本質的な動作を基本的な演算子を用いて記述することができる．ＣＣＳは，互いに独立した動作体(agent)同士が通信を行いながら，並行して計算を進めるようなシステムをモデルとしている．ＣＣＳにおける計算は，動作体間の通信によって進行する．それぞれの通信には名前がつけられており，ＣＣＳの記述する動作体は，通信につけられた名前によって表現される振舞い(behaviour)によって同定される．動作体を記述する表現は動作式(behaviour expression)として定義されている．

このＣＣＳの応用として，ＯＳや分散処理における並行処理の概念やプログラミング言語自体をＣＣＳに変換し，ＣＣＳのモデルと対応づけることで，形式的な体系のもとに並行計算をモデル化することができる[1(chapter 9),4,5,6,13]．このとき問題となる点は，動作体間の通信を計算の基本的なメカニズムとした場合，ＣＣＳのモデルの制約が非常に厳しいことである．これは，ＣＣＳにおける通信の名前づけが，動作式に対して静的に決まっていなければならないことである．このことは，ＣＣＳと同様なモデルを持つＣＳＰ[8]においても指摘されている弱点である[7]．このため，ＣＣＳを暗黙的に拡張して，通信の名前づけを動的に変化させることが行われている[4,5,13]．しかし，これは元のＣＣＳの枠組みを越えるものである．

本稿では，このような問題点に対して，通信の動的な名前づけを許すようにＣＣＳを明示的に拡張する．元のＣＣＳの名前づけが定数であるのに対して，この拡張では変数を含むような項による通信の名前づけを導入する．名前を表す項に含まれる変数は，動作体間の通信による値の受け渡しによって具体化される．拡張した動作式の振舞いは，名前を表す項から通信の名前の領域への写像を決めることによって定義する．このように拡張した動作式の意味は，振舞いが同じであるようなＣＣＳの動作式と関係づけることができる．ここで，項から名前の領域への写像は，名前を表す項のトップレベルの関数記号に割り当てられる関数によって決定する．

このような動的な名前づけを導入した動作式を定式化するために，次の２点を新たに導入した概念として提案する．

(1) 制限演算

制限演算は，ＣＣＳにおいてモジュール構造を表現するために通信の名前を隠蔽する演算である．動的な名前づけを行うときには，隠蔽すべき通信の名前が変化するので，拡張が必要である．本稿では，名前を表す項のトップレベルに現れる関数記号による制限を提案する．

(2) 名前割り当て

名前を表す項のトップレベルに現れる関数記号に対して，変数を含まない項を通信の名前の領域へ写像する関数を割り当てることを名前割り当てと呼び，独立した概念として扱う．この割り当てを変化させることで，項で表された通信の等価性を変化させることができる．

直観的には，項による通信の名前づけを許した動作式は，高級プログラミング言語で書かれたプログラムに対応し，これに名前割り当てを施して得られる動作式は，そのプログラムの実行における振舞いを表していると考えられる．ここで，名前割り当ては，並行処理系に対応づけることができる．

本稿の構成は以下の通りである．２章では，Milnerが定義したＣＣＳを説明する．３章でＣＣＳの拡張を行って，動作が同じであるようなＣＣＳの動作式と対応づけることで，拡張した動作式の意味を定める．４章では，３章で行った拡張に対する問題点について議論を行い，今後の課題を挙げる．５章ではまとめを述べる．

2. 準備

2.1 CCS

以下に，ＣＣＳの基本的な諸定義を列挙する．

【定義】値表現(value expression)

値表現Eは以下の２つのものから構成される項である．

(1) 変数　x, y, ...

(2) 関数記号　f1, f2, ...

ここで，関数記号に対してはアリティが唯一に決まっているものとする．特に，引数が0個である関数記号に続く括弧は省略して，例えば，a(), b()をそれぞれa, bと表すことにする．変数を含まない値表現を，値(Value)といい，値の集合をＶで，その要素をvで表す．値表現における関数記号は，値の上の全域関数(total function)を表す．値表現のタプルも値表現であるとし，$\tilde{E}$ と書く．（空タプルは'<>'で表す．）また，変数，値のタプルはそれぞれ $\tilde{x}$, $\tilde{v}$で表す．　■

【定義】名前集合(names)，名前余集合(co-names)

名前集合Δは固定されたひとつの集合である．その要素はギリシャ小文字$\alpha, \beta, \gamma, \ldots$で表す．名前余集合は以下のような全単射$\overline{(\cdot)}$によってΔから決まる集合であり，$\overline{\Delta}$で表す．（ここで，$\Delta \cap \overline{\Delta} = \phi$）

$$\overline{(\cdot)}: \alpha \mapsto \overline{\alpha} \quad (\alpha \in \Delta,\ \overline{\alpha} \in \overline{\Delta})$$

この関数を相補関数という．$\overline{\alpha}$を名前αの余名(co-name)と呼ぶ．さらに，$\overline{\overline{\alpha}} = \alpha$が成立する．　■

【定義】ラベル集合(labels)，ソート(sort)

ラベル集合Λは，$\Lambda = \Delta \cup \overline{\Delta}$であり，その要素をλで表す．$\overline{\lambda}$はλの相補ラベル(complementary label)であるという．Λに属さない特別なシンボルτが存在し，τも含めてラベルの要素を表す記号にはμを用いる．（$\mu \in \Lambda \cup \{\tau\}$）さらに，ラベル集合においては，$(\Lambda \cup \{\tau\}) \cap V = \phi$であるとする．また，Δの要素を正ラベル，$\overline{\Delta}$の要素を負ラベルとよぶ．Λの部分集合をソートと呼ぶ．　■

以上の定義は，次の１箇所を除いて文献[1(chapter 5)]の引用である．文献[1]の定義では，ラベル集合と値表現の集合との関係に制約は存在しないが，ここではこの２つの集合に共通な要素が存在しないことを仮定している．

ここで，記法を簡単にするために，これらの定義に基づいて，名前集合とラベル集合の間に以下の２つの関数name，namesを定義する．

$$\text{name}: \lambda \mapsto \begin{cases} \lambda & \text{if } \lambda \in \Delta \\ \overline{\lambda} & \text{if } \lambda \in \overline{\Delta} \end{cases} \quad (\text{ここで，} \lambda \in \Lambda)$$

$$\text{names}: L \mapsto \{\text{name}(\lambda); \lambda \in \Lambda\} \quad (\text{ここで，} L \subseteq \Lambda)$$

【定義】ラベル替え(relabelling)

ソートＬとＭの間の関数Ｓ：Ｌ→Ｍが次の２条件を満たすとき，Ｓはラベル替えであるという．

(1) 全単射である．

(2) 相補関数を保存する．

すなわち，$\overline{S(\alpha)} = S(\overline{\alpha})$　■

【定義】動作識別子(behaviour identifier)

動作識別子bは，予め次の３つが割り当てられているシンボルである．

(1) アリティ（bの引数の個数をn(b)で表す．$n(b) \geq 0$）

(2) ソート　（bのソートをL(b)で表す．）

(3) 動作識別子の集合　（ID(b)で表す．）　■

各動作識別子の意味は，後述するような動作識別子宣言で定義される．

【定義】基本動作式(basic behaviour expression)

基本動作式B_θは次のようなBNFで与えられる式である．（ＣＣＳの演算子'|'との混同を避けるため，以下にあげるすべてのBNFでは選択を表す記号として'|'の代わりに'#'を用いる．）

B_θ::=NIL

\# B_θ + B_θ

\# $\alpha\tilde{x}.B_\theta$　（$\alpha \in \Delta$，$\tilde{x}$は変数のタプル）

\# $\overline{\alpha}\tilde{E}.B_\theta$　（$\alpha \in \Delta$，$\tilde{E}$は値表現のタプル）

\# $\tau.B_\theta$

\# B_θ | B_θ

\# B_θ[S]　（Sはラベル替え）

\# $B_\theta \setminus A$　（$A \subseteq \Delta$）

if bb then Be else Be　（bbはブール式）
b(E1,...,Em)　（bは動作識別子，m=n(b)，E1,...,Emは値表現）

Be\Aにおいて，特にAが単一集合{α}であるときには，Be\αと書くことにする．■

基本動作式Beに自由に現れる変数の集合をFV(Be)で表し，Beのソートを L(Be)によって定義する．また，Be中に現れる動作識別子の集合をID(Be)で表す．FV(Be),L(Be),ID(Be)はBeのBNF定義に沿って，以下のように定義される．

(1) Be = NIL のとき
FV(Be)=φ　L(Be)=φ　ID(Be)=φ

(2) Be = Be1 + Be2 のとき
FV(Be)=FV(Be1)∪FV(Be2)
L(Be)=L(Be1)∪L(Be2)
ID(Be)=ID(Be1)∪ID(Be2)

(3) Be = α x̃.Be1 のとき　FV(Be)=FV(Be1)-Var(x̃)
/*ここで，Var(x̃)は変数のタプルx̃に現れる変数の集合を表す．*/
L(Be)=L(Be1)∪{α}　ID(Be)=ID(Be1)

(4) Be = ᾱẽ.Be1 のとき　FV(Be)=FV(Be1)∪FV(ẽ)
L(Be)=L(Be1)∪{ᾱ}　ID(Be)=ID(Be1)

(5) Be = τ.Be1 のとき
FV(Be)=FV(Be1)　L(Be)=L(Be1)
ID(Be)=ID(Be1)

(6) Be = Be1 | Be2 のとき
FV(Be)=FV(Be1)∪FV(Be2)
L(Be)=L(Be1)∪L(Be2)
ID(Be)=ID(Be1)∪ID(Be2)

(7) Be = Be1[S] のとき
FV(Be)=FV(Be1)　L(Be)=S(L(Be1))
ID(Be)=ID(Be1)

(8) Be = Be1\A のとき
FV(Be)=FV(Be1)　L(Be)=L(Be1)-(A∪Ā)
ID(Be)=ID(Be1)

(9) Be = if bb then Be1 else Be2 のとき
FV(Be)=FV(bb)∪FV(Be1)∪FV(Be2)
L(Be)=L(Be1)∪L(Be2)
ID(Be)=ID(Be1)∪ID(Be2)

(10) Be = b(E1,...,Em) のとき
FV(Be)=∪i FV(Ei)
L(Be)=L(b)　ID(Be)={b}

基本動作式に現れる動作識別子は次のような動作識別子宣言によって，その意味が（通常は再帰的に）定められている．

【定義】動作識別子宣言
(behaviour identifier declaration)
動作識別子宣言Dは，次のようなBNFで与えられる．
D ::= ε　/* 空系列 */
De {; De }
De::= b(x1,...,xm)⇐Be　/* m=n(b),Beは動作式 */
各Deを基本動作識別子宣言と呼ぶ．■

動作識別子宣言Dに対しても，FV(D),L(D),ID(D)が以下のように定義される．
(1) D = ε のとき　FV(D)=φ　L(D)=φ　ID(D)=φ
(2) D = D1; ... ;Dn のとき　FV(D)=∪i FV(Di)
L(D)=∪i L(Di)　ID(D)=∪i ID(Di)

ここで，各基本動作識別子宣言Di=(bi(xi1,...,ximi)⇐Bei)において，
FV(Di)=FV(Bi)-{Xi1,...,Ximi}
L(Di)=L(bi)（⊇L(Bi)）
ID(Di)=ID(Bei)∪{bi}

また，以下の議論において記法を簡単化するため，biを定義する基本動作式BeiをBody(bi)と書くことにする．

【定義】動作式(behaviour expression)
動作式Bは，基本動作式Beと動作識別子宣言Dの対<Be,D>である．（ここで，ID(Be)⊆ID(D)）
動作式Bは，FV(Be)=FV(D)=φを満たすとき閉じている(closed)という．動作式全体の集合をBで表すことにする．■

2．2　動作関係による意味論

ＣＣＳの意味論は動作式上に導出関係を定めることによって与えられる[1(chapter 5)]．ここにおける導出関係は，動作関係(action relation)[2]と呼ばれ，ラベルづけされた矢印 $\xrightarrow{com}$ ⊆Ｂ×Ｂで表すことにする．ここで，com∈{(λ,ṽ);λ∈Λ,ṽは値のタプル}∪{τ}

comの各要素を取り出す関数label,valueを次のように定める．

$$label(com)=\begin{cases}\lambda & \text{if } com=(\lambda,\tilde{v})\\ \tau & \text{if } com=\tau\end{cases}$$

$$value(com)=\begin{cases}\tilde{v} & \text{if } com=(\lambda,\tilde{v})\\ ANY & \text{if } com=\tau\end{cases}$$

以下に定める動作関係において，その動作識別子宣言は不変である．したがって，以後，動作式は基本動作式だけで表し，そこに現れる動作識別子の意味は，対となる動作識別子宣言によって決まっているものとする．
動作関係は次の規則を満たす最小の関係である．

(1)　動作
$\alpha\tilde{x}.B \xrightarrow{com} B\{value(com)/\tilde{x}\}$　(label(com)=α)
$\bar{\alpha}\tilde{v}.B \xrightarrow{com} B$　(label(com)=ᾱ)
$\tau.B \xrightarrow{com} B$　(com=τ)

(2) 和 (summation)
（以下の規則の記法では，横線の上が含意の前件，下が後件を表す．）

$$\frac{B1 \xrightarrow{com} B1'}{B1+B2 \xrightarrow{com} B1'+B2}$$

$$\frac{B1 \xrightarrow{com} B1'}{B2+B1 \xrightarrow{com} B2+B1'}$$

(3) 合成 (composition)

$$\frac{B1 \xrightarrow{com} B1'}{B1 \mid B2 \xrightarrow{com} B1' \mid B2}$$

$$\frac{B1 \xrightarrow{com} B1'}{B2 \mid B1 \xrightarrow{com} B2 \mid B1'}$$

$$\frac{B1 \xrightarrow{com} B1' \quad B2 \xrightarrow{com'} B2' \quad label(com)=label(com')}{B1 \mid B2 \xrightarrow{\tau} B1' \mid B2'}$$

(4) 制限 (restriction)

$$\frac{B \xrightarrow{com} B' \quad (name(label(com)) \notin A)}{B \setminus A \xrightarrow{com} B' \setminus A}$$

(5) ラベル替え (relabelling)

$$\frac{B \xrightarrow{com} B'}{B[S] \xrightarrow{S(com)} B'[S]}$$

ここで $S(com)=\begin{cases}(S(label(com)), value(com)) & \text{if } label(com) \in \Lambda \\ \tau & \text{if } com=\tau\end{cases}$

(6) 動作識別子 (identifier)

$$\frac{B_b\{v1/x1,\ldots,vm/xm\} \xrightarrow{com} B' \quad (b \in ID(D1), body(b)=B_b, m=n(b))}{b(v1,\ldots,vm) \xrightarrow{com} B'}$$

(7) 条件文 (conditional)

$$\frac{B1 \xrightarrow{com} B1'}{\underline{if}\ true\ \underline{then}\ B1\ \underline{else}\ B2 \xrightarrow{com} B1'}$$

$$\frac{B2 \xrightarrow{com} B2'}{\underline{if}\ false\ \underline{then}\ B1\ \underline{else}\ B2 \xrightarrow{com} B2'}$$

ＣＣＳは以上のような意味論のもとで，直観的には，図１に示すように通信を行いながら並行に計算を進めるようなシステムのモデルを表している．ここで，正ラベルに関する動作は値を入力する通信を，負ラベルに関する動作は値を出力する通信を表す．値は互いに相補な関係にある正ラベルと負ラベルの間で受け渡される．ＣＣＳにおけるプログラムは次のように定義されている．

【定義】プログラム[1(chapter 5)]
プログラムは閉じた動作式である． ■

【命題１】[1(chapter 5)]
Bがプログラムであり，$B \xrightarrow{com} B'$ のとき，B'もプログラムである． □

すなわち，プログラムは導出関係に関して閉じている．
以下の議論では，プログラム，すなわち，閉じた動作式のみを対象とする．

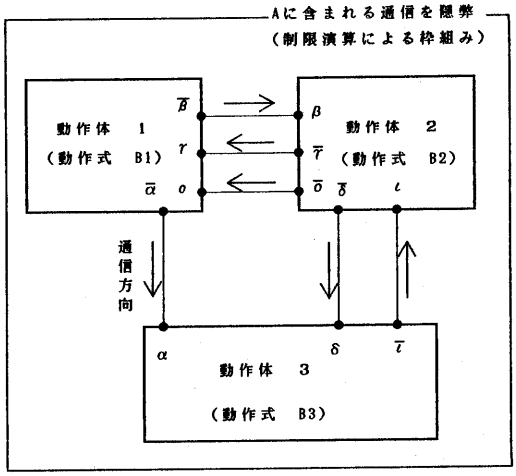

図１ ＣＣＳの直観的なモデル（(B1 | B2 | B3)＼Aの場合）

３．ＣＣＳの拡張

３．１ ＣＣＳの問題点

ＣＣＳによって，以上のように相補ラベル間の通信という単純なメカニズムに基づいて，抽象的なレベルで並行計算をモデル化することができる．しかし，ＣＣＳでは，動作式の意味が静的に与えられている．動作は，動作式が与えられたときに決定される静的な集合における選択から決定される．しかし，実際の並行処理システムのモデルとＣＣＳのモデルとを対応づけようとするとき，このことは非常にきびしい制約であると考えられる．実際的なプログラミング言語や並行処理における基本的な概念をＣＣＳに翻訳して，その計算メカニズムをモデル化しようとするときには，通信の名前は実行時に決定するような枠組みで記述する方が自然である．このため，通信の名前にパラメータを許し，パラメータの値を通信によって受け渡すことによって，その通信より後に起こる通信の名前を決定できることを許すような枠組みを提案する．

以下の議論では，このパラメータを持つ名前を変数を含むような項で表し，次のようにＣＣＳを拡張する．まず，名前が項のままであるような動作式を動作前表現として定義する．次に，項に現れるトップレベルの関数記号に値から名前への関数を割り当てることによって，拡張動作式を定義する．この拡張されたＣＣＳの意味は，同じ振舞いをするような（拡張前の）ＣＣＳの意味に対応づけられる．

3．2　動作前表現

【定義】名前表現

名前表現は次の形をした表現である．

f(E1,...,Em)　(m≧0)

ここで，fは名前関数記号，E1,...,Emは値表現である．それぞれのfに対してアリティは唯一に決まっているものとし，引数の個数mをn(f)と書く．（厳密に引数のタイプまで考慮する場合のアリティはarity(f)と書く．）名前表現全体の集合をH，その要素をηで表す．また，特に引数がすべてVの要素である（引数の数が0個のときも含める）ような名前表現の集合をΞで表し，その要素はξで表す．　■

Hに対しても，Δと同様に全単射の相補関数$\overline{(\cdot)}$"が定義される．Hの各要素に相補関数を施して得られた集合を$\overline{H}$"で表す．

$\overline{(\cdot)}$" : $\eta \mapsto \overline{\eta}$"　($\eta \in H$, $\overline{\eta}$"$\in \overline{H}$")

$\overline{(\cdot)}$と同様に，$\overline{(\cdot)}$"∘$\overline{(\cdot)}$"は恒等関数とする．

【定義】ラベル表現

ラベル表現は$H \cup \overline{H}$"の要素であり，νで表す．Hに属するラベル表現を正ラベル表現，$\overline{H}$"に属するラベル表現を負ラベル表現と呼ぶことにする．　■

τに対応して，Hにも$\overline{H}$"にも属さないシンボル$\underline{\tau}$が存在する．

記法を簡単にするため，ラベル表現に対して次のような関数を定義する．

fname: $\nu \mapsto$f

(ν=f(E1,...,Em) または $\overline{f(E1,...,Em)}$"　(m=n(f)))

fnames: $\Theta \mapsto \{fname(\theta); \theta \in \Theta\}$　($\Theta \subseteq H \cup \overline{H}$")

【定義】ソート

$H \cup \overline{H}$"の部分集合に現れる名前関数記号の集合をソートと呼ぶ．　■

【定義】ラベル替え

2つのソートL,Mにおいて，LからMへの関数Sが次の条件を満たすとき，Sはラベル替えであるという．また，Sはラベル表現f(E1,...,Em)に対して，S(f(E1,...,Em))=S(f)(E1,...,Em)のように拡張される．

(1) Sは全単射

(2) ラベル表現上の相補関数を保存する．

すなわち，$S(\overline{\eta})$"=$S(\overline{\eta}")$

(3) 名前関数記号のアリティを保存する．

すなわち，arity(f)=arity(S(f))　■

ラベル表現を導入するときは，動作識別子を次のように定義する．

【定義】動作識別子

動作識別子bは，次の3つが予め割り当てられたシンボルである．

(1) アリティ　（bの引数の個数はn(b)で表す．）

(2) ソート　（ソートはL(b)で表す．）

(3) 動作識別子の集合　（ID(b)で表す）　■

MilnerのCCSでは，動作を同定する名前が定数であるのに対して，ここで行う拡張では変数を含むような項によって動作を同定する．項で表現することにより，実行時に受け渡される値によって，動的に名前を決定することができる．次に名前表現に基づいた動作式を，動作前表現として定義する．

【定義】基本動作前表現

(basic behaviour pre-expression)

基本動作前表現B_θ'は，次のようなBNFによって与えられる．

B_θ' ::= NIL
　# B_θ' + B_θ'
　# $\eta \tilde{x}.B_\theta$'
　# $\overline{\eta}$"$\tilde{E}.B_\theta$'
　# $\underline{\tau}.B_\theta$'
　# B_θ' | B_θ'
　# B_θ'[S]　/*Sはラベル替え*/
　# B_θ'\F　/*Fはラベル関数記号の集合*/
　# if bb then B_θ' else B_θ'
　# b(E1,...,Em)　/*bは動作識別子*/

B_θ'\Fにおいて，とくにFが単一集合{f}のときには，B_θ'\fと書くことにする．　■

B_θ'の自由変数の集合FV(B_θ')およびソートL(B_θ')とB_θ'に現れる動作識別子の集合ID(B_θ')を次のように定義する．

(1) FV(NIL)=ϕ　L(NIL)=ϕ　ID(NIL)=ϕ

(2) FV(B_θ'1+B_θ'2)=FV(B_θ'1)∪FV(B_θ'2)
　　L(B_θ'1+B_θ'2)=L(B_θ'1)∪L(B_θ'2)
　　ID(B_θ'1+B_θ'2)=ID(B_θ'1)∪ID(B_θ'2)

(3) FV($\eta \tilde{x}.B_\theta$')=(FV(B_θ')∪FV(η))-Var($\tilde{x}$)
　　L($\eta \tilde{x}.B_\theta$')=L(B_θ')∪{fname(η)}
　　ID($\eta \tilde{x}.B_\theta$')=ID(B_θ')

(4) FV($\overline{\eta}$"$\tilde{E}.B_\theta$')=FV(B_θ')∪FV($\overline{\eta}$")∪FV($\tilde{E}$)
　　L($\overline{\eta}$"$\tilde{E}.B_\theta$')=L(B_θ')∪{fname($\overline{\eta}$")}
　　ID($\overline{\eta}$"$\tilde{E}.B_\theta$')=ID(B_θ')

(5) FV($\underline{\tau}.B_\theta$')=FV($B_\theta$')
　　L($\underline{\tau}.B_\theta$')=L($B_\theta$')
　　ID($\underline{\tau}.B_\theta$')=ID($B_\theta$')

(6) FV(B_θ'1 | B_θ'2)=FV(B_θ'1)∪FV(B_θ'2)
　　L(B_θ'1 | B_θ'2)=L(B_θ'1)∪L(B_θ'2)
　　ID(B_θ'1 | B_θ'2)=ID(B_θ'1)∪ID(B_θ'2)

(7) FV(B_θ'[S])=FV(L(B_θ'))
　　L(B_θ'[S])=S(L(B_θ'))
　　ID(B_θ'[S])=ID(B_θ')

(8) FV(B_θ'\F)=FV(B_θ')
　　L(B_θ'\F)=L(B_θ')-F
　　ID(B_θ'\f)=ID(B_θ')

(9) FV(if bb then B_θ'1 else B_θ'2)
=FV(bb)∪FV(B_θ'1)∪FV(B_θ'2)
L(if bb then B_θ'1 else B_θ'2)
=L(B_θ'1)∪L(B_θ'2)
ID(if bb then B_θ'1 else B_θ'2)
=ID(B_θ'1)∪ID(B_θ'2)
(10) FV(b(E1,...,Em))=∪i FV(Ei)
L(b(E1,...,Em))=L(b)
ID(b(E1,...,Em))={b}

【定義】動作識別子宣言
動作前表現における動作識別子宣言D'は次のように与えられる.
D' ::= ε /*空系列*/
D_θ' {; D_θ' }
D_θ' ::= b(x1,....,xm)⇐B_θ' /*B_θ'は動作前表現*/
各D_θ'を基本動作識別子宣言と呼ぶ. ■

動作識別子宣言D'における自由変数の集合FV(D'), ソートL(D'), 動作識別子の集合ID(D')は, 以下のように定義される.
(1) FV(ε)=φ L(ε)=φ ID(ε)=φ
(2) D=D1',...,Dn', 基本動作識別子宣言をDi'=bi(xi1,...,xim$_i$)⇐B_θ'i (i=1,...,n)とするとき
FV(D1';...;Dn')=∪i FV(Di')
L(D1';...;Dn')=∪i L(Di')
ID(D1';...;Dn')=∪i ID(Di')
ここで, 各基本動作識別子宣言Di'において, FV(Di')=FV(B_b'i)-{xi1,...,xim$_i$}, L(bi)⊇L(B_bi'), ID(Di')=ID(B_b'i)∪{bi}

【定義】動作前表現(behaviour pre-expression)
動作前表現B'は, 基本動作前表現B_θ'と動作識別子宣言D'の対<B_θ',D'>である. (ここで, ID(B_θ')⊆ID(D')) 閉じた動作前表現は, FV(B_θ')=FV(D')=φであるような動作前表現である. また, 動作前表現の全体集合はBpreで表す. ■

3. 3 名前割当て

動作前表現の名前関数記号に対して, 名前関数を割り当てることで拡張動作式を定義する.

【定義】名前割当て(name assignment)
名前割当てgは, 名前関数記号を名前関数に割り当てる全域関数である. さらに, gによって, $\underline{\tau}$はτに, $\overline{(\cdot)}^n$は$\overline{(\cdot)}$に割り当てられる.
g : f ↦ ((v1,...,vm) ↦ α)
ここで, f∈fnames(H) m=n(f) α∈Δ
g : $\underline{\tau}$ ↦ τ
g : $\overline{(\cdot)}^n$ ↦ $\overline{(\cdot)}$
gは, 名前表現に対して次のように拡張される.
η=f(E1,...,Em)に対し, g(η)=g(f)(E1,...,Em) ■

【定義】拡張動作式

拡張動作式Bexは, gによって閉じた動作前表現の名前関数記号に名前関数を割り当てて得られる表現である. 名前割当てgのもとで動作前表現B'から得られる拡張動作式を$[\![B']\!]_g$で表す. (以下, 混同のおそれがないかぎり$[\![B']\!]_g$のgは省略する.)
動作前表現から拡張動作式は, 以下のように定義される.

(1) [[NIL]]=NIL
(2) [[B_θ'1+B_θ'2]]=[[B_θ'1]]+[[B_θ'2]]
(3) [[η※.B_θ']]=g(η)※[[B_θ']]
(4) [[$\overline{\tau}^n$※.B_θ']]=g($\overline{\tau}^n$)※[[B_θ']]
(5) [[$\underline{\tau}$.B_θ']]=τ.[[B_θ']]
(6) [[B_θ'1 | B_θ'2]]=[[B_θ'1]] | [[B_θ'2]]
(7)
[[B_θ'[S]]]=
- NIL (if B_θ'=NIL)
- [[B_θ'1[S]+B_θ'2[S]]] (if B_θ'=B_θ'1+B_θ'2)
- g(S(η))※.[[B_θ'1[S]]] (if B_θ'=η※.B_θ'1)
- g(S($\overline{\tau}^n$))※.[[B_θ'1[S]]] (if B_θ'=$\overline{\tau}^n$※.B_θ'1)
- τ.[[B_θ'1[S]]] (if B_θ'=$\underline{\tau}$.B_θ'1)
- [[B_θ'1[S] | B_θ'2[S]]] (if B_θ'=B_θ'1 | B_θ'2)
- [[B_θ'1[T]]] (T=S∘S') (if B_θ'=B_θ'1[S'])
- [[B_θ'1[S]\S(f)]] (if B_θ'=B_θ'1\f)
- if bb then [[B_θ'1[S]]] else [[B_θ'2[S]]] (if B_θ'=if bb then B_θ'1 else B_θ'2)

(8)
[[B_θ'\F]]=
- NIL (if B_θ'=NIL)
- [[B_θ'1\F+B_θ'2\F]] (if B_θ'=B_θ'1+B_θ'2)
- g(η)※.[[B_θ'1\F]]\g(η) (if B_θ'=η※.B_θ'1, fname(η)∈F)
- g(η)※.[[B_θ'1\F]] (if B_θ'=η※.B_θ'1, fname(η)∉F)
- g($\overline{\tau}^n$)※.[[B_θ'1\F]]\g($\overline{\tau}^n$) (if B_θ'=$\overline{\tau}^n$※.B_θ'1, fname($\overline{\tau}^n$)∈F)
- g($\overline{\tau}^n$)※.[[B_θ'1\F]] (if B_θ'=g($\overline{\tau}^n$)※.B_θ'1, fname($\overline{\tau}^n$)∉F)
- τ.[[B_θ'1\F]] (if B_θ'=$\underline{\tau}$.B_θ'1)
- [[B_θ'1 | B_θ'2]]\(∪i range(g(fi))) (if B_θ'=B_θ'1 | B_θ'2, fi∈F) /*range(g(fi))は名前関数g(fi)の値域を表す. i=1,...,n */
- if bb then [[B_θ'1\F]] else [[B_θ'2\F]] (if B_θ'=if bb then B_θ'1 else B_θ'2)

(9) [[if bb then B_θ'1 else B_θ'2]]
=if bb then [[B_θ'1]] else [[B_θ'2]]
(10) [[b'(E1,...,Em)]]=b(E1,...,Em)
ここで, b'の識別子宣言をb'(x1,...,xm)⇐B_b'とする

とき，bの識別子宣言は，$b(x1,\ldots,xm)\Leftarrow[\![B_b']\!]$
拡張動作式の全体集合はB_{ex}で表す． ■

拡張動作式に対して，FV, L, IDをそれぞれつぎのように定める．

(1) $FV(NIL)=\phi \quad L(NIL)=\phi \quad ID(NIL)=\phi$

(2) $FV([\![B_\theta'1+B_\theta'2]\!])=FV([\![B_\theta'1]\!])\cup FV([\![B_\theta'2]\!])$
$L([\![B_\theta'1+B_\theta'2]\!])=L([\![B_\theta'1]\!])\cup L([\![B_\theta'2]\!])$
$ID([\![B_\theta'1+B_\theta'2]\!])=ID([\![B_\theta'1]\!])\cup ID([\![B_\theta'2]\!])$

(3) $FV([\![\eta\tilde{x}.B_\theta']\!])=FV(g(\eta))\cup(FV([\![B_\theta']\!])-Var(\tilde{x}))$
$L([\![\eta\tilde{x}.B_\theta']\!])=fname(\eta)\cup L([\![B_\theta']\!])$
$ID([\![\eta\tilde{x}.B_\theta']\!])=ID([\![B_\theta']\!])$

(4) $FV([\![\overline{\eta}^{*}\tilde{E}.B_\theta']\!])=FV(g(\overline{\eta}^{*}))\cup Var(\tilde{E})\cup FV([\![B_\theta']\!])$
$L([\![\overline{\eta}^{*}\tilde{E}.B_\theta']\!])=fname(\overline{\eta}^{*})\cup L([\![B_\theta']\!])$
$ID([\![\overline{\eta}^{*}\tilde{E}.B_\theta']\!])=ID([\![B_\theta']\!])$

(5) $FV([\![\tau.B_\theta']\!])=FV([\![B_\theta']\!])$
$L([\![\tau.B_\theta']\!])=L([\![B_\theta']\!])$
$ID([\![\tau.B_\theta']\!])=ID([\![B_\theta']\!])$

(6) $FV([\![B_\theta'1\mid B_\theta'2]\!])=FV([\![B_\theta'1]\!])\cup FV([\![B_\theta'2]\!])$
$L([\![B_\theta'1\mid B_\theta'2]\!])=L([\![B_\theta'1]\!])\cup L([\![B_\theta'2]\!])$
$ID([\![B_\theta'1\mid B_\theta'2]\!])=ID([\![B_\theta'1]\!])\cup ID([\![B_\theta'2]\!])$

(7) $FV([\![B_\theta'[S]]\!])= FV([\![B_\theta']\!])$
$L([\![B_\theta'[S]]\!])=S(L([\![B_\theta']\!]))$
$ID([\![B_\theta'[S]]\!])=ID([\![B_\theta']\!])$

(8a) $FV(g(\eta)\tilde{x}.[\![B_\theta'\setminus f]\!]\setminus g(\eta))$
$=FV(g(\eta))\cup(FV([\![B_\theta'\setminus f]\!])-Var(\tilde{x}))$
$L(g(\eta)\tilde{x}.[\![B_\theta'\setminus f]\!]\setminus g(\eta))=L([\![B_\theta'\setminus f]\!])\cup fname(\eta)$
$ID(g(\eta)\tilde{x}.[\![B_\theta'\setminus f]\!]\setminus g(\eta))=ID([\![B_\theta'\setminus f]\!])$

(8b) $g(\eta)\tilde{x}.[\![B_\theta'\setminus f]\!]$
(8c) $g(\overline{\eta}^{*})\tilde{E}.[\![B_\theta'1\setminus f]\!]\setminus g(\overline{\eta}^{*})$ } についても同様
(8d) $g(\overline{\eta}^{*})\tilde{E}.[\![B_\theta'1\setminus f]\!]$

(9) $FV([\![\underline{if}\ bb\ \underline{then}\ B_\theta'1\ \underline{else}\ B_\theta'2]\!])$
$=Var(bb)\cup FV([\![B_\theta'1]\!])\cup FV([\![B_\theta'2]\!])$
$L([\![\underline{if}\ bb\ \underline{then}\ B_\theta'1\ \underline{else}\ B_\theta'2]\!])$
$=L([\![B_\theta'1]\!])\cup L([\![B_\theta'2]\!])$
$ID([\![\underline{if}\ bb\ \underline{then}\ B_\theta'1\ \underline{else}\ B_\theta'2]\!])$
$=ID([\![B_\theta'1]\!])\cup ID([\![B_\theta'2]\!])$

以上のように拡張動作式において，変数の束縛は，正ラベル表現のみによって行われる．また，動作式と同様に，$FV([\![B']\!])=\phi$のとき，拡張動作式$[\![B']\!]$は閉じているという．

【命題２】
動作前表現B'が閉じているとき，任意のgに対し，拡張動作式$[\![B']\!]_\theta$は閉じている． □

以下では，閉じた拡張動作式のみを扱う．

３．４ 拡張動作式における動作関係

閉じた拡張動作式は動作式と同様の動作関係 $\xrightarrow{com}$ $\subseteq B_{ex}\times B_{ex}$（ここで，com=$(\lambda,\tilde{v})$または$\tau$）によって意味を与えることができる．拡張動作式における和，合成，動作識別子，条件文に対する動作関係の規則は，動作式のそれと同じである．（ただし，動作識別子宣言にもgによる割り当てを行う．）また，拡張動作式の構成から，ラベル替えの規則は存在しない．以下に，和，合成，条件文以外の規則を定義する．

(1') 動作

$$g(\xi)\tilde{x}.Bex \xrightarrow{com} Bex\{value(com)/\tilde{x}\}\quad (label(com)=g(\xi))$$
$$g(\tilde{\xi})\tilde{v}.Bex \xrightarrow{com} Bex\quad (label(com)=g(\tilde{\xi}))$$
$$\tau.Bex \xrightarrow{com} Bex\quad (com=\tau)$$

(4') 制限

$$\frac{Bex \xrightarrow{com} Bex' \quad (name(label(com))\neq g(\theta),\ \theta\in\Xi\cup\tilde{\Xi})}{B\setminus g(\theta) \xrightarrow{com} B'\setminus g(\theta)}$$

(6') 動作識別子

動作前表現の動作識別子宣言に
$b(x1,\ldots,xm)\Leftarrow B_b$が存在するとする．

$$\frac{[\![B_b\{v1/x1,\ldots,vm/xm\}]\!] \xrightarrow{com} B' \quad (b\in ID(D1),\ body(b)=B_b,\ m=n(b))}{b(v1,\ldots,vm) \xrightarrow{com} B'}$$

３．５ 拡張動作式と動作式の関係

動作式上の動作関係 $\xrightarrow{com}$ のもとで，次のような合同関係が定義される．この合同関係は，内部通信であるτも含めた動作関係に対して定義されていることから，強合同関係(strong congruence)[1(chapter5)]と呼ばれる．

【定義】動作式上の強合同関係～
関係～は次のように帰納的に定義される．
(1) B1 $\sim_\theta$ B2は，常に成立する．
(2) B1 $\sim_{k+1}$ B2
iff 任意のcomに対して，
(i) B1 $\xrightarrow{com}$ B1' のとき，
$\exists$B2'. B2 $\xrightarrow{com}$ B2'かつB1' $\sim_k$ B2'
(ii) B2 $\xrightarrow{com}$ B2' のとき，
$\exists$B1'. B1 $\xrightarrow{com}$ B1'かつB1' $\sim_k$ B2'
(3) B1 $\sim$ B2 iff $\forall k\geq 0$. B1 $\sim_k$ B2 ■

【補題３】 ～は，同値関係である． □

拡張動作式の上にも，～と同様な関係を定義する．

【定義】拡張動作式上の関係～ex
関係～exは次のように帰納的に定義される．
(1) Bex1 $\sim_{ex\ \theta}$ Bex2は常に成立する．
(2) Bex1 $\sim_{ex\ k+1}$ Bex2
iff 任意のcomに対して，

(i) Bex1 $\xrightarrow{com}$ Bex1' のとき,
$\exists$Bex2'. Bex2 $\xrightarrow{com}$ Bex2' かつ Bex1 $\sim ex_k$ Bex2
(ii) Bex2 $\xrightarrow{com}$ Bex2' のとき,
$\exists$Bex1'. Bex1 $\xrightarrow{com}$ Bex1' かつ Bex1 $\sim ex_k$ Bex2
(3) Bex1 $\sim ex$ Bex2 iff $\forall k \geq 0$. Bex1 $\sim ex_k$ Bex2 ■

【補題４】 $\sim ex$は，同値関係である． □

【命題５】
$\sim ex$は，拡張動作式において合同関係である． □

次に，動作式の動作関係 $\xrightarrow{com}$ と拡張動作式の動作関係 $\xrightarrow{com}$ とを次のよう対応づけるような関係を定義する．

【定義】
関係$\simeq \subseteq$ Bex×Bは次のように帰納的に与えられる．
(1) Bex $\simeq_0$ Bは常に成立する．
(2) Bex $\simeq_{k+1}$ B
iff
任意のcomに対し，
(i) Bex $\xrightarrow{com}$ Bex' のとき，
$\exists$B'. B $\xrightarrow{com}$ B' かつ Bex' $\simeq_k$ B'
(ii) B $\xrightarrow{com}$ B' のとき，
$\exists$Bex'. Bex $\xrightarrow{com}$ Bex' かつ Bex' $\simeq_k$ B'
(3) Bex $\simeq$ B iff $\forall k \geq 0$. Bex $\simeq_k$ B ■

以上のようにして，通信の名前づけを項に拡張したＣＣＳの動作式は，項における関数記号（名前関数記号）に実際の名前を対応させる関数を割り当てることで，もとＣＣＳの動作式と対応づけることができる．
この対応づけに対して，次のような性質が成り立つ．

【補題６】 $\forall k \geq 0. \simeq_{k+1} \subseteq \simeq_k$

証明：kに関する帰納法． □

【定理７】
Bex$\simeq$B iff $\forall$com.
(i) Bex $\xrightarrow{com}$ Bex' のとき，
$\exists$B'. B $\xrightarrow{com}$ B', Bex' $\simeq$ B'
(ii) B $\xrightarrow{com}$ B' のとき，
$\exists$Bex'. Bex $\xrightarrow{com}$ Bex', Bex' $\simeq$ B' □

【定理８】 $\simeq \circ \sim = \sim ex \circ \simeq$

証明：（$\subseteq$）$\sim ex$では，対称律が成立するので，$\sim ex = (\sim ex)^{-1}$．よって$\sim ex \circ \simeq \circ \sim \subseteq \simeq$ を証明すればよい．右辺の$\simeq$の定義におけるkに関する帰納法で証明する．
いま，Bex $\sim ex$ Bex', Bex' $\simeq$ B', B' $\sim$ Bとする．
(I) k=0 のときは自明
(II) Bex $\xrightarrow{com}$ Bex1のとき，Bex $\sim ex$ Bex'より
$\exists$Bex1'. Bex' $\xrightarrow{com}$ Bex1' かつ Bex1 $\sim ex$ Bex1'
また，Bex' $\simeq$ B'より
$\exists$B1'. B' $\xrightarrow{com}$ B1' かつ Bex1' $\simeq$ B1'
また，B' $\sim$ Bより
$\exists$B1. B $\xrightarrow{com}$ B1 かつ B1' $\sim ex$ B1
従って，帰納法の仮定からBex1$\simeq_k$B1となるB1が存在する．
B $\xrightarrow{com}$ B1のときも同様にして，
$\exists$Bex1. Bex1 $\sim ex$ B1
よってBex $\simeq_{k+1}$ B
(III) $\sim ex \circ \simeq \circ \sim \subseteq \simeq$
（$\supseteq$）
$\sim$では対称律が成立することから，
$\simeq \subseteq \sim ex \circ \simeq \circ \sim$ □

この定理は，τによる動作も考慮した動作関係から得られる同値類が，同じ通信の振舞いを対応づけることで，拡張動作式と動作式において一致していることを示している．（図２）

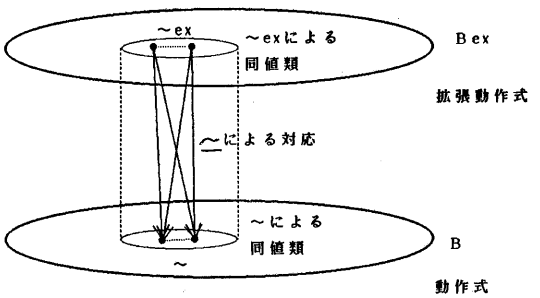

図２ 拡張動作式と動作式との対応

４．議論

４．１ 動作前表現における動作関係

動作前表現にも同様に，動作関係によって直接意味を定めることができる．動作前表現における動作識別子宣言は，次に定義する動作関係によっては変化しない．したがって，以後，動作前表現は，その基本動作前表現で表すことにし，それに伴う動作識別子宣言が存在して，個々の動作識別子はその意味が決まっているものとする．
動作前表現における動作関係は以下の規則を満たす最小の関係である．動作前表現における動作関係は
$\xrightarrow{com} \subseteq$ Bpre×Bpre で表すことにする．（ここで com$\in$ {(ξ, $\bar{v}$)}$\cup$ {$\underline{\tau}$}）

(1") 動作

$\xi \overset{\sim}{\times}$. B $\xrightarrow{com}$ Bσ
ここで，com=(ξ, $\bar{v}$)，σは$\bar{v}=\sigma$を満たす代入
$\xi \overset{\sim}{\forall} \bar{v}$. B $\xrightarrow{com}$ B
ここで，com=(ξ, $\bar{v}$)，σは$=^{E}\sigma$を満たす代入
$\underline{\tau}$. B $\xrightarrow{\tau}$ B

(2") 和

$$\frac{B1 \xrightarrow{com} B1'}{B1+B2 \xrightarrow{com} B1'+B2}$$

$$\frac{B1 \xrightarrow{com} B1'}{B2+B1 \xrightarrow{com} B2+B1'}$$

(3") 合成

$$\frac{B1 \xrightarrow{com} B1'}{B1 \mid B2 \xrightarrow{com} B1' \mid B2}$$

$$\frac{B1 \xrightarrow{com} B1'}{B2 \mid B1 \xrightarrow{com} B2 \mid B1'}$$

$$\frac{B1 \xrightarrow{com} B1,\ B2 \xrightarrow{com'} B2'}{B1 \mid B2 \xrightarrow{\tau} B1' \mid B2'}$$

ここで, com=(ξ, v̄)のとき, com'=(ξ", v̄)

(4") 制限

$$\frac{B \xrightarrow{com} B' \quad (fname(label(com)) \notin F)}{B \setminus F \xrightarrow{com} B' \setminus F}$$

(5") ラベル替え

$$\frac{B \xrightarrow{com} B'}{B[S] \xrightarrow{S(com)} B'[S]}$$

ここで, $S(com)=(S(label(com)), label(com))$
$S(f(\bar{v}))=S(f)(\bar{v})$

(6") 動作識別子

$$\frac{B_b\{v1/x1, \ldots, vm/xm\} \xrightarrow{com} B_b' \quad (body(b)=B_b, m=n(b))}{b(v1, \ldots, vm) \xrightarrow{com} B_b'}$$

(7") 条件文

$$\frac{B1 \xrightarrow{com} B1'}{\underline{if}\ true\ \underline{then}\ B1\ \underline{else}\ B2 \xrightarrow{com} B1'}$$

$$\frac{B2 \xrightarrow{com} B2'}{\underline{if}\ false\ \underline{then}\ B1\ \underline{else}\ B2 \xrightarrow{com} B2'}$$

動作前表現においても次のようにプログラムを定義する.

【定義】プログラム
<u>プログラム</u>は閉じた動作前表現である. ■

【命題９】
プログラムは動作関係 $\xrightarrow{com}$ において閉じている. □

以上のような動作前表現の直接の意味論と, 名前割り当てを行った後で対応づけられる動作式の意味論の対応については今後の課題である. ただ, 前章で扱ったようなτに関する動作関係を陽に示す意味づけでは, 動作式の分類が細かすぎると思われる. 例えば, 次のような動作前表現B'に名前割り当てgを行ったとする.

$B' = (a\langle\rangle.NIL) \mid (\bar{b}\langle\rangle.NIL)$
$g = \{a \mapsto \alpha, b \mapsto \alpha\}$

$B' \xrightarrow{\tau}$ B"となる動作前表現B"は存在しないにもかかわらず, $[\![B']\!] \xrightarrow{\tau}$ C となる動作式Cとして NIL | NIL が存在する. ただし, gで割り当てられる名前関数の値域の重なりが無い場合[13]は直接対応づけることができる.

4. 2 名前割り当てにおける順序

名前割り当てgから, 次のような関数$(g^{-1})^{-1}$を作る.

$(g^{-1})^{-1}(\alpha)$
$=\{f(v1, \ldots, vm)\ ;\ g(f)(v1, \ldots, vm)=\alpha \in \Delta\}$

$(g^{-1})^{-1}(\alpha)$は, gによって名前αに割り当てられる名前表現の集合を返す. これを用いて, 名前割り当てに対して, 次のような順序の導入が考えられる.

【定義】
名前割り当てg1, g2に対して, つぎのような半順序$\sqsubseteq$を導入する.
$g1 \sqsubseteq g2$ <u>iff</u> $\forall \alpha \in \Delta.\ (g1^{-1})^{-1}(\alpha) \supseteq (g2^{-1})^{-1}(\alpha)$ ■

$g1 \sqsubseteq g2$であることは, 直観的に, g1による割り当ての方がg2に比べて実際に通信が行われる名前が, 割り当て前の表現に対して潰れていることを示している. 動作前表現において, 異なる通信で表されていても, 名前割り当てを行うことにより, その区別がつかなくなる名前表現がg1による割り当ての方が多い.

このようにgを可変にした理由は, 実際の並行処理系において以下のような対応があるからである. 高級プログラミング言語で, ユーザが並行プログラムを書くとき, 実際の通信がどのように行われるかは直接意識されないのが普通であると思われる. そのような高級プログラミング言語の処理系において, スケジューリングや, メモリのプロセス割当のように, 何らかの意味で, gによる名前割り当てに対応する操作をシステムプログラムが暗黙的に行っていると考えることができる. ユーザの書く並行プログラムのもつ意味を動作前表現の意味に対応させたとき, システムはできるだけ, 粗い(小さい)名前割り当てのもとに実行する方がよいと思われる. このことが, 名前割り当てを導入して, その上に上記のような順序を導入した動機である. gの細かさと, gによって動作前表現から得られる拡張動作式の動作関係との対応については, 今後の課題である.

4．3　制限演算の扱い

本稿で行ったようなＣＣＳの拡張は，他に文献[3,4,5]でも行われているが，[5]では，制限演算が取り扱われていない．また，[3]では，動作式を与える時点で，可能性のある全ての制限を列挙しておく必要がある．動的な名前づけを行った場合，一般的には，制限するラベルも動的に変化させることが必要になる．本稿で行った拡張では通信が行われる際の名前関数記号によって，制限を行っている．B_{α}'1\fの場合，単に，fに割り当てられる名前関数g(f)の値域すべてを制限することが考えられるが，これは通信を制限し過ぎると思われる．このため，拡張動作式の動作関係の定義において，名前関数記号が一致する場合のみ，制限を行うようにしている．しかし，合成された前表現　B1 | B2 を関数記号fによって制限する場合(B1 | B2)\fは，内部通信を許すため，割り当てられた名前関数g(f)の値域となるすべての名前で制限している．ここでは実際的な面から，このように ad hoc に定義した．制限演算に関する扱いは一般的に複雑であり[10,11]，動作前表現と拡張動作式の対応づけにおいてさらに検討を要する．

5．　まとめ

ＣＣＳにおける通信の名前づけを，定数から変数を含む項に拡張することで，実行時に通信の名前が決定できるように拡張を行った．ここで定義した拡張動作式は，名前を表す項のトップレベルに現れる関数記号に，値と名前を対応づける関数を割り当てることによって得られる．通信を実際にどのように決定するかは，この名前割り当てによって与えられる．名前割り当てを行って得られる拡張動作式の意味はＣＣＳの動作式に対応して与えられる．

実際の並行処理システムとの対応では，名前に項を含むような動作式はユーザプログラムに対応し，その項に対して実際にどのような通信を割り当てて実行させるかという決定は，コンパイラ，インタプリタ，ＯＳなどのシステムプログラムによる並行処理系に対応していると考えられる．

名前を項のまま持つような動作前表現に直接与えた動作関係と，名前割り当てによって得られる拡張動作式の動作関係との対応づけを行うことは，今後の課題である．

謝　辞

日頃ご指導賜る豊橋技術科学大学本多波雄学長，中京大学福村晃夫教授，ならびに御討論頂いた平田富夫講師，直井徹助手をはじめとする研究室の皆様に感謝します．

参考文献

[1] R.Milner:"A Calculus of Communicating Systems", Lecture Notes in Computer Science 92 (1980)
[2] R.Milner:"Lectures on a Calculus for Communicating Systems", Lecture Notes in Computer Science 197, pp.199-220 (1983)
[3] R.Milner:"Calculi for Synchrony and Asynchrony", Theoretical Computer Science 25, pp.267-310 (1983)
[4] S.A.Smolka and R.E.Strom:"A CCS Semantics for NIL", IBM Journal of Research and Development 31, pp.556-570 (1987)
[5] T.W.Doeppner Jr. and A. Glacalone:"A Formal Description of the UNIX Operating System", Proc of the 2nd Annual ACM Symposium on Principle of Distributed Computing pp.241-253,(1983)
[6] M.C.B.Hennessy and W. Li:"Translating a Subset of Ada into CCS", Foramal Description of Programming Cocepts II (Bjoerner ed.), pp.227-249 (1983)
[7] N.Francez:"Extended Naming Conventions for Communicating Processes", Science of Computer Programming 3, pp.101-114 (1983)
[8] C.A.R.Hoare:"Communicating Sequential Processes", Prentice-Hall (1985)
[9] P.B.Hansen:"The Architecture of Concurrent Programs", Prentice-Hall (1977)
[10] S.D.Brooks, C.A.R.Hoare and A.W.Roscoe:"A Theory of Communicating Sequential Processes", JACM 31, pp.560-599 (1984)
[11] E.-R. Olderog and C.A.R.Hoare:"Specification-Oriented Semantics for Communicating Processes", Acta Informatica 23, pp.9-66 (1986)
[12] "Occamプログラミングマニュアル", 啓学出版 (1984)
[13] 結縁・坂部・稲垣:"ＣＣＳに基づくモニタの動作の形式的記述",信学技報 COMP87-84 (1988)