情報処理学会論文誌　プログラミング　Vol.5　No.2　79–87 (Mar. 2012)

物流システム記述のための多重Ambient Calculus

樋口 昌宏[1,a)]　加藤 暢[1]

受付日 2011年9月30日, 採録日 2011年12月30日

概要：本論文ではプロセス式の組により物流システムを記述する多重 Ambient Calculus を提案する．多重 Ambient Calculus では，各プロセス式中の名前は共通名と個別名に分けられる．共通名を持つアンビアントに関する遷移はそれらの名前を共有するプロセスで同時に発生するという制約を多重 Ambient Calculus の遷移規則に導入することでプロセス間の同期を表現する．本論文では多重 Ambient Calculus の構文規則，遷移規則を定義し，多重プロセス間の弱双模倣等価関係について述べるとともに，それらに関する基本的な諸性質について述べる．また，いくつかの記述例を通して，多重 Ambient Calculus により，個々の貨物の取扱いについて個別のプロセス式として記述することができることを示す．そのような記述により，通常の Ambient Calculus と比較して物流システムの持つ対称性，並行性をより適切に表現できるとともに，貨物の追加削除の際の式の変更作業も容易になる．

キーワード：仕様記述，プロセス代数，移動型プロセス，弱双模倣等価

The Multiple Ambient Calculus for Specifying Freight Systems

Masahiro Higuchi[1,a)]　Toru Kato[1]

Received: September 30, 2011, Accepted: December 30, 2011

Abstract: We are investigating the way for describing freight systems in the Ambient Calculus and constructing freight management systems based on it. Since a freight system is modeled by a single huge formula in pure Ambient Calculus, it is inadequate to express the concurrency of freight systems and it should be tedious work to modify the formula. To overcome such shortcomings, we propose the Multiple Ambient Calculus, which enables us to model freight systems by a set of formulae. In the Multiple Ambient Calculus, the names are classified into global and individual ones, and the reduction rule is defined to ensure that the reduction concerning to global names must happen simultaneously among the processes. In this paper, we define the syntax and reduction rules of the Multiple Ambient Calculus, introduce a weak bi-simulation relation between the Multiple Ambient processes, and show some basic properties on the calculus. We also give an example of multiple ambient formula expressing a freight system such that each sub-formula expresses the handling of one cargo. Using the formula, we will explain that the formula expresses enough concurrency among cargo handlings, and we can modify the formula only by adding a sub-formula in case of adding one cargo handling to the freight system.

Keywords: formal specification, process algebra, process mobility, weak bisimulation

1. はじめに

Ambient Calculus [1] は，Microsoft Research の Luca Cardelli と Andrew D. Gordon によって開発されたプロセス代数であり，動的な階層構造を持つシステムを形式的に記述することができる言語である．この特徴を活用した種々の応用が研究されている [2], [3], [4].

海上物流などでの貨物の取扱い量が増大し，貨物の管理あるいは物流システムそのものの監視ないしは管理の重要性が高まってきており，物流業界ではさまざまな形での物流管理の方法が研究されている [5], [6], [7]．一方では，物流の大規模化にともない，物流管理システムの信頼性の向上が重要になってきている．そのような物流管理システム

[1] 近畿大学
Kinki University, Higashiosaka, Osaka 577–8502, Japan
[a)] higuchi@info.kindai.ac.jp

を構築する基盤として，物流計画を記述するための形式的な枠組みが重要であると考えられる．そのような枠組みが確立できれば，物流計画の厳密な定義が行えるため，解釈の違いによる取り違えのような事態を防止できるとともに，物流計画そのものが所期の性質を満たしていることを形式的に検証するといったことが可能となる．

そのような形式的枠組みの 1 つとして，我々は Ambient Calculus の動的に変動するオブジェクトの階層構造を形式的に記述できるという特徴に着目し，Ambient Calculus による物流計画の記述とそれに基づく物流監視システムの構築を提案している [8]．提案手法は，物流計画を Ambient Calculus 式によって記述し，実際の貨物の取扱いが，記述した式の遷移に沿ったものになっているかを逐次確認することで，計画どおりの貨物の取扱いが行われているかどうかを監視しようとするものである．さらに，そのようにして記述された物流計画に対してモデル検査を行うことにより，物流計画そのものが所期の性質を満たしていることを形式的に検証することができるモデル検査システムを開発している [9]．

しかしながら，通常の Ambient Calculus による物流計画の記述では，1 つのプロセス式により物流全体を記述するため，1 つの貨物の取扱いに関する記述が式中のさまざまな場所に散在するような書き方をせざるをえず，プロセス式が複雑な形のものになりがちで，記述性，了解性が十分ではなかった．また同様の理由から，同じ手順で取り扱われる貨物が多数存在するといった物流が持っている対称性や，それらの取扱いは任意の順序で行われてもよいといった並行性を十分に表現することが困難であった．さらには，取り扱う貨物が 1 つ追加されたり，削除されたりする際にも式全体を修正する必要があるという問題があった．

そこで本論文では複数のプロセス式により物流計画を記述する多重 Ambient Calculus を提案する．多重 Ambient Calculus では，各プロセス式中の名前は共通名と個別名に分ける．そして，共通名を持つアンビアントに関する遷移はそれらの名前を共有するプロセスで同時に発生する，という制約を多重 Ambient Calculus の遷移規則に導入することでプロセス間の同期を表現する．多重 Ambient Calculus では，物流計画を複数のプロセス式に分けて記述することができるため，1 つのプロセス式では 1 つの貨物の取扱いのみを記述するという記述スタイルが可能になる．そのようなスタイルで物流を記述すれば，名前のみが異なり同様の取扱いを要求する貨物が複数存在することを直接的に表現できたり，貨物の追加削除の際の式の修正が格段に容易になったりすることが期待される．以降では，多重 Ambient Calculus の構文規則，遷移規則を定義し，多重プロセス間の弱双模倣等価関係について述べるとともに，それらに関する基本的な性質を示す．また，いくつかの記述例を通して，多重 Ambient Calculus により，個々の貨物の取扱いについて個別のプロセス式として記述することができることを示す．

2. Ambient Calculus と物流記述

2.1 移動プリミティブを持つ Ambient Calculus

文献 [1] において，通信プリミティブを含む Ambient Calculus のサブクラスとして，移動プリミティブのみを備えた Ambient Calculus が定義されており，チューリング機械を模倣できるという意味で万能であることが示されている．本論文では，物流システムのモデル化のために移動プリミティブのみを備えた Ambient Calculus を利用することを考える．以下にその構文規則およびプロセス式間の基本的な合同関係（構造合同）の定義を示す．

定義 2.1（構文規則）

n	names	$M ::=$	capabilities
$P, Q ::=$	processes	$in\ n$	can enter n
$(\nu n)P$	restriction	$out\ n$	can exit n
0	inactivity	$open\ n$	can open n
$P \mid Q$	composition		
$!P$	replication		
$n[P]$	ambient		
$M.P$	action		

□

通常，プロセス式中の「(」,「)」および「0」は混乱のない範囲で省略される．

定義 2.2（構造合同）

$P \equiv P$	(Refl)
$P \equiv Q \Rightarrow Q \equiv P$	(Symm)
$P \equiv Q,\ Q \equiv R \Rightarrow P \equiv R$	(Trans)
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Res)
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	(Par)
$P \equiv Q \Rightarrow\ !P \equiv\ !Q$	(Repl)
$P \equiv Q \Rightarrow n[P] \equiv n[Q]$	(Amb)
$P \equiv Q \Rightarrow M.P \equiv M.Q$	(Action)
$P \mid Q \equiv Q \mid P$	(Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Par Assoc)
$!P \equiv P \mid\ !P$	(Repl Par)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Res Res)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$	
if n が P の自由変数でない	(Res Par)
$(\nu n)(m[P]) \equiv m[(\nu n)P]$	
if $n \neq m$	(Res Amb)
$P \mid 0 \equiv P$	(Zero Par)
$(\nu n)0 \equiv 0$	(Zero Res)
$!0 \equiv 0$	(Zero Repl)

□

定義 2.2 および以降では「⇒」は含意を表す．

以降では互いに構造合同なプロセス式は同一視することとし，特に (Repl Par) より「$!P \equiv\ !P \mid \underbrace{P \mid \ldots \mid P}_{n}$」

($1 \leq n$) が得られるが，「$!P$」をそれらの標準形と考え，「$!P \mid P \mid \ldots \mid P$」のような式は考慮の対象としない．

Ambient Calculus では，ambient 構文「$n[P]$」を用いて内部にプロセスを持つオブジェクト（以降アンビアントと呼ぶ）を表現することができ，これを再帰的に用いることによりアンビアント間の入れ子構造を表現できる．たとえば $l[m[n[0]]]$ によりアンビアント l の中にアンビアント m が存在し，さらにその中にアンビアント n が存在するというような階層構造を表現できる．以降ではアンビアント l の中にアンビアント m が存在するとき，l が m の親，m が l の子であるという．親子関係の反射推移閉包を祖先子孫関係という．

アンビアント内に記述されたケーパビリティにより，階層構造の動的な変動を記述できる．ケーパビリティを消費することにより，プロセス式は異なる階層構造を持つ式に遷移する．式の遷移規則を以下に示す．

定義 2.3（遷移規則）

$$n[in\ m.P \mid Q] \mid m[R] \to m[n[P \mid Q] \mid R] \quad \text{(In)}$$
$$m[n[out\ m.P \mid Q] \mid R] \to n[P \mid Q] \mid m[R] \quad \text{(Out)}$$
$$open\ n.P \mid n[Q] \to P \mid Q \quad \text{(Open)}$$
$$P \to Q \Rightarrow (\nu n)P \to (\nu n)Q \quad \text{(Res)}$$
$$P \to Q \Rightarrow n[P] \to n[Q] \quad \text{(Amb)}$$
$$P \to Q \Rightarrow P \mid R \to Q \mid R \quad \text{(Par)}$$
$$P' \equiv P, P \to Q, Q \equiv Q' \Rightarrow P' \to Q' \quad \text{(Cong)} \qquad \square$$

遷移規則 (In) は，n アンビアントが自身の持つケーパビリティ「$in\ m$」を消費し，m アンビアントの中に入る遷移を表す．遷移規則 (Out) は，n アンビアントがケーパビリティ「$out\ m$」を消費し，m アンビアントの外へ出る遷移を表す．また，遷移規則 (Open) は，ケーパビリティ「$open\ n$」が消費されることにより n アンビアントが消滅し，n が内部に保持していたプロセスはその親に継承される遷移を表している．「$\to$」の反射推移閉包を「$\overset{*}{\to}$」で表す．

遷移規則の定義では ambient 構文「$n[P]$」を用いて記述されたアンビアントであっても，action 構文「$M.P$」を用いて何らかのケーパビリティに先行されているもの，およびその子孫のアンビアントは遷移に関与しない．そのようなアンビアントを非活性といい，そうでないものを活性ということにする．

Ambient Calculus では，「$Name \triangleq P$」というようなプロセス定義式によりプロセス式に名前を与えておいて，別の（自身でもよい）プロセス式中に $Name$ と書くことでプロセス式を呼び出すことができる．一般には再帰的なプロセス定義式も許される．

2.2　物流システムの記述例

動的に変動するアンビアント間の階層構造を記述できるという Ambient Calculus の特徴を利用して，物流システムを表現できる．これを簡単な記述例を用いて説明する．

例 2.4　船 SHIP が東京港（TK）でコンテナヤード（CY）からコンテナ CT を積み込んで神戸港（KB）へ輸送する物流計画を，以下の式により表現することができる．

```
SHIP[ in TK.
          (load[out SHIP. in CY. in CT]
         | open lcomp. out TK. in KB)]
| TK[ CY[ CT[open load. out CY. in SHIP.
            lcomp[out CT] ] ] ]
| KB[ CY[] ]
```

この式からケーパビリティと，ケーパビリティに先行されるプロセス式をすべて削除して得られる式

```
SHIP[] | TK[ CY [ CT[] ] ] | KB[ CY[] ]
```

は，船（SHIP）アンビアント，東京港（TK）アンビアントと神戸港（KB）アンビアントが並行して存在し，TK と KB はそれぞれ内部にコンテナヤード（CY）アンビアントを持ち，TK 内の CY にはコンテナ（CT）アンビアントが 1 つ存在するという，物流システムの現在の状態を表していると考えることができる．また，上記の式は

```
TK[ SHIP[ load[out SHIP. in CY. in CT]
        | open lcomp. out TK. in KB]
  | CY[ CT[ open load. out CY. in SHIP.
            lcomp[out CT] ] ] ]
| KB[ CY[] ]
```

という式に遷移可能で，その遷移は船が東京港に入港したという動作を表現していると考えることができる．この遷移により船が入港したこと，すなわち積み込みが可能になったことをコンテナに知らせる役割を持つ load アンビアントが活性化され，load アンビアントの一連の遷移により，

```
TK[ SHIP[open lcomp. out TK. in KB]
  | CY[ CT[open load. out CY. in SHIP.
            lcomp[out CT]
          | load[] ] ] ]
| KB[ CY[] ]
```

という式に遷移する．ここで CT アンビアントは open load ケーパビリティを消費することにより，引き続く荷物の積み込みを表す遷移 out CY と in SHIP が可能となり，

```
TK[ SHIP[open lcomp. out TK. in KB
        | CT[lcomp[out CT] ] ]
  | CY[] ]
| KB[ CY[] ]
```

へ遷移する．さらにこの式では積み込み完了を船に伝える lcomp アンビアントが活性化される．lcomp アンビアントが CT アンビアントを出ることにより SHIP の open lcomp ケーパビリティを消費可能となり，引き続く東京港からの出港，神戸港への入港を表す遷移 out TK. in KB が行われ，

```
TK[ CY[] ]
```

```
| KB[ SHIP[CT[]] | CY[] ]
```

へ遷移する．□

このように必要な同期をとりながらモノを輸送する物流計画を，アンビアントの階層構造の遷移列として表現することができる．以降では上記式のコンテナ名，積荷港名，荷降ろし港名を変数にし，さらに荷降ろし港からの出港まで記述した以下のプロセス定義式を引用することにする．この式は簡単のため z での荷降ろしについて記述していないため不完全なものであるが，通常の海上輸送に用いられる送り状 [10], [11]（invoice）の記載内容を表現したものと見なすことができる．

定義 2.5（Invoice 式）

```
Invoice(x, y, z) ≜
  SHIP[ in y. (load[out SHIP.in CY. in x]
             | open lcomp. out y. in z. out z)]
| y[ CY[ x[open load. out CY. in SHIP.
          lcomp[out x] ] ] ]
| z[ CY[] ]                                    □
```

同様に，以降の便宜のため，東京港，神戸港，門司港（MJ）の順に寄港する SHIP の航路のみを表現した式を定義しておく．

定義 2.6（SHIProute 式）

```
SHIProute ≜
  SHIP[ in TK. out TK. in KB. out KB.
        in MJ. out MJ]
| TK[] | KB[] | MJ[]                           □
```

2.3　物流システム記述のための Ambient Calculus

物流システム記述の便宜を図るために，Ambient Calculus にいくつかの定義を追加するとともに，いくつかの制限を課す．

一般のプロセス式 P から活性アンビアント間の階層構造のみを表すプロセス式を抽出する関数 $\mathcal{H}$ を，以下のように定義する．

定義 2.7（アンビアント階層の抽出）

$\mathcal{H}((\nu n)P) = (\nu n)(\mathcal{H}(P))$	restriction
$\mathcal{H}(0) = 0$	inactivity
$\mathcal{H}(P \mid Q) = \mathcal{H}(P) \mid \mathcal{H}(Q)$	composition
$\mathcal{H}(!P) = 0$	replication
$\mathcal{H}(n[P]) = n[\mathcal{H}(P)]$	ambient
$\mathcal{H}(M.P) = 0$	action

□

$\mathcal{H}(P)$ は，プロセス式 P が表す物流システムの現在の状態を表現しているものと考えることができる．そのような $\mathcal{H}(P)$ に現れるアンビアントは船やコンテナなどの物理的なモノを表すものであるか，例 2.4 で用いた load や lcomp のような何らかの役割を持ったインスタンス化されたソフトウェアオブジェクトを表すもののいずれかである．どちらにしても，何らかの手順を経て実体化されたものなので，$\mathcal{H}(P)$ が無限となるような P は物流システムを表現しているとは考えられない．一方，通常の Ambient Calculus では再帰的プロセス定義を許しているため，$P \triangleq n[P]$ というようなプロセス定義式を書くことにより簡単に無限の階層構造を持つプロセスが定義できてしまい，これも不適当である．

本研究では複製や再帰により無限に展開可能なプロセス式は排除しないが，$\mathcal{H}(P)$ が無限になるようなプロセス式 P は議論の対象としない．そのため，プロセス式定義に以下の制約を付加することとする．

定義 2.8（プロセス定義式の制約）

プロセス定義式 $Name \triangleq P$ において，右辺の P が（相互再帰を含む）再帰的なプロセス呼び出し $Name'$ を含む場合，それらの再帰呼び出しは $M.Name'$ の形で何らかのケーパビリティに先行されなければならない．□

この制限をプロセス定義式の記述に課することにより，アンビアント階層の定義 $\mathcal{H}(M.P) = 0$ より，再帰呼び出しの部分はアンビアント階層を求める際には無視され，結果として任意のプロセス式 P に対して $\mathcal{H}(P)$ は有限となる．

便宜上プロセス定義式の展開は式の遷移に影響を与えるとき，すなわち，プロセス呼び出しが活性アンビアント中でケーパビリティに先行されていない場合のみ展開することとし，「$n[in\ m.Name]$」のような式では $Name$ はそのまま放置することにする．

例 2.4 では TK と KB 内に同じ名前の CY アンビアントがあるが，これらは別のものを表していると考えるべきである．このため本論文では同じ名前の活性アンビアントが存在する場合，一意の ID を持ち区別可能であると考える．

定義 2.9（アンビアント ID）

プロセス式 P 中に同じ名前のアンビアントが存在する場合，活性であるかどうかにかかわらず，あらかじめ何らかの ID が与えられているものとする．一方，複製演算子やプロセス呼び出しで動的に生成されるアンビアントについては複製が生成されたとき，もしくは展開されたときに ID が付与されるものとする．$\mathcal{H}$ によりアンビアント階層を抽出する場合も，ID 付きの階層が抽出されるものとする．□

表記上，必要であればアンビアント ID はアンビアント名に添字として記述することとし，あらかじめ与えられる ID の空間と複製生成時，定義式展開時に付与される ID の空間は互いに素であるとする．

ただし，遷移規則でケーパビリティとマッチングがとられるアンビアント名については ID なしの名前を対象とすることができる．たとえば $c[in\ SHIP] \mid SHIP_1[\,] \mid SHIP_2[\,]$ なる式において $SHIP_1$，$SHIP_2$ を同じ名前を持ち，異なる ID を持つアンビアントと考えると，どちらも c アンビアントの「$in\ SHIP$」ケーパビリティとマッチング可能で，$SHIP_1[c[\,]] \mid SHIP_2[\,]$ と $SHIP_1[\,] \mid SHIP_2[c[\,]]$ の両方へ遷移可能である．

プロセス式の遷移によりアンビアント階層は変化する．アンビアント階層の変化を遷移ラベルで表現することにする．遷移規則ごとのラベリング規則を以下のように定める．

定義 2.10（遷移ラベル）

$n[in\ m.P \mid Q] \mid m[R] \rightarrow_l m[n[P \mid Q] \mid R]$,
　$l =$ “n enter m”　(In)
$m[n[out\ m.P \mid Q] \mid R] \rightarrow_l n[P \mid Q] \mid m[R]$,
　$l =$ “n exit m”　(Out)
$open\ n.P \mid n[Q] \rightarrow_l P \mid Q$,
　$l =$ “n disappear”　(Open)　□

遷移ラベル中のアンビアント名は ID 付きのものとする．このため

$$!(n[in\ m \mid P]) \mid m[] \rightarrow !(n[in\ m \mid P]) \mid m[n[P]]$$

のような複製の生成をともなう遷移の際には，遷移前に複製が必要な数だけ生成され（上記の場合 $n[in\ m \mid P]$ が 1 個），それとともにそれらの複製中の活性化されるアンビアント（上記の例では n，および P を単独のプロセス式と見た場合の P 中での活性アンビアント）に ID が与えられ，遷移ラベル中の n は与えられた ID を含むものと考える．

$P_0 \rightarrow_{l_1} P_1$，$P_1 \rightarrow_{l_2} P_2$，$\cdots$，$P_{n-1} \rightarrow_{l_n} P_n$（$n \geq 0$）であるとき $P_0 \stackrel{*}{\rightarrow}_{l_1 l_2 \ldots l_n} P_n$ と書くことにする．

Ambient Calculus で物流システムのような状態遷移システムをモデル化する場合，遷移ラベルはそのシステムで生起したイベントを表現していると見なすことができる．以降では遷移ラベルとイベントを同一視することにする．

3. 多重 Ambient Calculus

文献 [8] で示した Ambient Calculus による物流記述は，単一のプロセス式で物流計画全体を記述するものであった．これに対して，多重 Ambient Calculus では n 個のプロセス式の組 $\overline{P} = (P_1, \ldots, P_n)$ で 1 つの物流計画全体を表現することを考える．以下では個々のプロセス式 P_i を個別式と呼び，$\overline{P}$ を全体式と呼び，全体式で表す物流計画全体をプロセス系と呼ぶことにする．

3.1 大域アンビアント

多重 Ambient Calculus では，アンビアント名の名前空間を，個々の個別式のみに関連する個別アンビアントのためのものと，一般に複数の個別式に共通に現れる大域アンビアントのためのものに分割する．全体式 $\overline{P}$ の大域アンビアント名の集合を $A_G(\overline{P})$ と書き，$\overline{P}$ の第 i 成分で用いられるアンビアント名の集合を $A^i(\overline{P})$，大域アンビアント名の集合を $A^i_G(\overline{P})$，個別アンビアント名の集合を $A^i_I(\overline{P})$ と書く．以降では，大域アンビアント名は先頭に大文字を用いて区別することとする．また一般性を失うことなく，1 つの全体式中の異なる個別式に同じ名前の個別アンビアントは存在しない（$A^i_I(\overline{P}) \cap A^j_I(\overline{P}) = \emptyset$）ものとする．また文脈から $\overline{P}$ が明らかなときには，$A_G(\overline{P})$，$A^i(\overline{P})$，$A^i_G(\overline{P})$，$A^i_I(\overline{P})$ は単に A_G，A^i，A^i_G，A^i_I と表記するものとする．

個別式のアンビアント階層 S から，指定したアンビアント名集合 X を持つもののみを取り出す射影関数 $\mathcal{G}_X$ を以下のように定義する．

定義 3.1（アンビアント名射影）

$\mathcal{G}_X((\nu n)S) = (\nu n)\mathcal{G}_X(S)$	restriction
$\mathcal{G}_X(0) = 0$	inactivity
$\mathcal{G}_X(S \mid T) = \mathcal{G}_X(S) \mid \mathcal{G}_X(T)$	composition
$\mathcal{G}_X(n[S]) = n[\mathcal{G}_X(S)]$, if $n \in X$	visible ambient
$\mathcal{G}_X(n[S]) = \mathcal{G}_X(S)$, if $n \notin X$	invisible ambient　□

定義 3.2（大域アンビアント階層）

全体式 $\overline{P} = (P_1, \ldots, P_n)$ について，$\mathcal{G}_{A_G}(\mathcal{H}(P_i)) = \mathcal{G}_{A^i}(H)$（$1 \leq i \leq n$）なる大域アンビアントのアンビアント階層 H が存在するとき，これを $\overline{P}$ の大域アンビアント階層といい，その集合を $\mathcal{H}_G(\overline{P})$ で表記する．　□

全体式の大域アンビアント階層は各個別式の大域アンビアント間の祖先子孫関係を継承したものである．このため大域アンビアント階層を求める際，各個別式の大域アンビアント間の祖先子孫関係の和集合をとり，さらにその推移閉包の関係を求めればよい．求めた関係が順序関係になっていなければ，個別式の間で大域アンビアントの祖先子孫関係に矛盾が含まれていることを意味しており，大域アンビアント階層は存在しない．求めた関係が順序関係であり，森（forest）の形になっている（A が B，C の共通の子孫なら，B，C 間にも祖先子孫関係が存在する）なら，大域アンビアント階層は一意である．

たとえば

```
(KB[SHIP[]] | MJ[],
 TK[] | MJ[] | SHIP[])
```

に対して，`KB[SHIP[]] | TK[] | MJ[]` はその式の唯一の大域アンビアント階層である．

一方，求めた関係が順序関係であり，森の形になっていない場合，大域アンビアント階層は一意には定まらない．たとえば

```
(KB[SHIP[]] | MJ[],
 TK[SHIP[]] | MJ[])
```

では共通の子孫 `SHIP` を持つ `KB`，`TK` の間に祖先子孫関係は存在せず，$\mathcal{G}_{A_G}(\mathcal{H}(P_i)) = \mathcal{G}_{A^i}(H)$ を満たす H として `TK[KB[SHIP[]]] | MJ[]` もしくは `KB[TK[SHIP[]]] | MJ[]` が存在し，大域アンビアント階層は一意ではない．

$\overline{P}$ とそこから個別式を 1 つ除去して得られる全体式の大域アンビアント階層について以下の性質が成り立つ．

補題 3.3　$\overline{P} = (P_1, \ldots, P_n)$ が大域アンビアント階層を持てば，$\overline{P^{-i}} = (P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n)$ も大域アンビアント階層を持つ．

[証明] $\mathcal{H}_G(\overline{P})$ 中の任意のアンビアント階層から，そのア

ンビアント階層の祖先子孫関係を保存するように P_i のみに現れるアンビアントを除去すれば，$\overline{P^{-i}}$ の大域アンビアント階層が得られる．　□

3.2　遷移規則

全体式 $\overline{P}$ に対して，遷移ラベルの集合を $\mathcal{L}(\overline{P})$ と書き，遷移ラベルに現れるアンビアントがともに（disappear の場合は 1 つ）$A^i(\overline{P})$ の要素であるような遷移ラベルの集合を $\mathcal{L}^i(\overline{P})$ と書く．また，それらがともに $A_G(\overline{P})$ の要素であるような遷移ラベルの集合を $\mathcal{L}_G(\overline{P})$ とし，そのようなラベルを持つ遷移を大域遷移という．また $\mathcal{L}^i_I(\overline{P}) = \mathcal{L}^i(\overline{P}) - \mathcal{L}_G(\overline{P})$ とし，そのようなラベルを持つ遷移を第 i 成分の個別遷移という．$\mathcal{L}(\overline{P})$，$\mathcal{L}_G(\overline{P})$，$\mathcal{L}^i(\overline{P})$，$\mathcal{L}^i_I(\overline{P})$ は文脈に応じて $\mathcal{L}$，$\mathcal{L}_G$，$\mathcal{L}^i$，$\mathcal{L}^i_I$ と略記する．

混乱のない範囲で，$l \notin \mathcal{L}^i$ のラベルを持つ遷移で第 i 成分が変化しない場合でも，$P_i \rightarrow_l P_i$ と書くこととする．

全体式の遷移規則を以下のように定める．

定義 3.4（全体式の遷移規則）

$$P_i \rightarrow_l Q_i \wedge l \in \mathcal{L}^i_I \wedge \mathcal{H}_G(\overline{P}) \neq \emptyset$$
$$\wedge \mathcal{G}_{A_G}(\mathcal{H}(P_i)) = \mathcal{G}_{A_G}(\mathcal{H}(Q_i))$$
$$\Rightarrow \overline{P} = (P_1, \ldots, P_i, \ldots, P_n) \rightarrow_l \overline{Q} = (P_1, \ldots, Q_i, \ldots, P_n) \quad \text{(individual)}$$

$$\forall i(\ P_i \rightarrow_l Q_i) \wedge l \in \mathcal{L}_G \wedge \mathcal{H}_G(\overline{P}) \neq \emptyset \wedge \mathcal{H}_G(\overline{Q}) \neq \emptyset$$
$$\Rightarrow \overline{P} = (P_1, \ldots, P_n) \rightarrow_l \overline{Q} = (Q_1, \ldots, Q_n) \quad \text{(global)}$$ □

たとえば

```
P̄ = ((KB[SHIP[out KB]] | TK[]),
     (TK[] | MJ[] | SHIP[]))
Q̄ = ((KB[] | TK[] | SHIP[]),
     (TK[] | MJ[] | SHIP[]))
```

に対して $\overline{P} \rightarrow_{\text{“}SHIP\ exit\ KB\text{”}} \overline{Q}$ である．

遷移条件に遷移元で大域アンビアント階層を持つことを条件としているので，大域アンビアント階層を持たない全体式からの遷移は存在しない．

補題 3.3 と定義 3.4 より，以下の性質が成り立つ．

補題 3.5　$(P_1, \ldots, P_n) \rightarrow_l (Q_1, \ldots, Q_n)$ $(l \notin \mathcal{L}^i_I)$ ならば，$(P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n) \rightarrow_l (Q_1, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_n)$ である．　□

補題 3.6　$(P_1, \ldots, P_n) \xrightarrow{*} (Q_1, \ldots, Q_n)$ ならば，$(P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n) \xrightarrow{*} (Q_1, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_n)$ である．　□

3.3　記述例

多重 Ambient Calculus で異なる貨物を 1 つの船で輸送する物流を記述した例を示す．

例 3.7　コンテナ m を東京（TK）から門司（MJ）へ，コンテナ n を神戸（KB）から門司へ，船 SHIP で輸送する物流を表す式を以下の 3 つの式で書くことができる．

```
(P1=Invoice(m, TK, MJ),
 P2=Invoice(n, KB, MJ),
 P3=SHIProute)
```

(P1, P2, P3) から可能な遷移についてみてみると，P1 と P3 で実行可能な “SHIP enter TK” のみであり，その結果

```
(  TK[ SHIP[ load[out SHIP. in CY. in m]
          | open lcomp. out TK. in MJ. out MJ]
     | CY[ m[ open load. out CY. in SHIP.
              lcomp[out m] ] ] ]
 | MJ[ CY[] ],
   Invoice(n, KB, MJ),
   TK[ SHIP[out TK. in KB. out KB. in MJ.
            out MJ] | KB[] | MJ[])
```

へ遷移する．ここで第 3 成分のみ見れば，SHIP exit TK が実行可能であるが，第 1 成分では SHIP アンビアントの out TK は open lcomp に先行されており，この遷移は実行可能でない．結局この後しばらく第 1 成分の個別遷移のみ実行可能で，例 2.4 でみたように，

```
(  TK[ SHIP[out TK. in MJ. out MJ | m[]]
     | CY[] ]
 | MJ[ CY[] ],
   Invoice(n, KB, MJ),
   TK[ SHIP[out TK. in KB. out KB. in MJ.
            out MJ] | KB[] | MJ[])
```

へと遷移する．ここで大域遷移 SHIP exit TK が可能になり，

```
(  SHIP[in MJ. out MJ | m[]]
 | TK[ CY[] ]
 | MJ[ CY[] ],
   Invoice(n, KB, MJ),
   SHIP[in KB. out KB. in MJ. out MJ]
 | TK[] | KB[] | MJ[])
```

へと遷移する．この後神戸港への入港，荷積み，出港に引き続いて門司港に入港し，

```
( TK[ CY[] ] | MJ[ SHIP[out MJ | m[]] | CY[] ],
  KB[ CY[] ] | MJ[ SHIP[out MJ | n[]] | CY[] ],
  TK[] | KB[] | MJ[ SHIP[out MJ]])
```

まで遷移する．この後，Invoice 式の簡略化のため記述されていない荷降ろし作業を行った後，SHIP は門司港を出港する　□

P1，P2 は前述のように送り状から得られた式であり，この 2 つの式だけでは東京と神戸の入港順序が指定されていない．船の航路表を表す式 P3 を追加することによって入港順序が指定されている．また，同じ船を用いて別の貨物を輸送する場合には，それらの貨物に関する Invoice 式を記述し，全体式に追加すればよい．

以上のように，この例では物流で実際に用いられている文書の記載内容を個別にプロセス式として記述し，それら

の組によって物流システムを記述できている．

4. 弱双模倣等価性

多重 Ambient Calculus による物流記述では，例 3.7 でみたように物流システムを表現した全体式 $\overline{P}$ に新たな個別式 Q を追加することで，より大きな物流を記述することを意図している．その際，追加した部分式 Q が $\overline{P}$ の振舞いを阻害しないか，あるいは $(\overline{P}, Q)$ が $\overline{P}$ の持つ望ましい性質を継承しているかどうかの議論が重要となる．ここではそのような議論のための枠組みとして全体式間の振舞いに関する等価関係を定義し，その関係について成立する性質について述べる．

4.1 全体式の弱双模倣等価性

2 つの全体式 $\overline{P}$，$\overline{Q}$ の振舞いを比較するうえで，それぞれにおいて生起可能なイベントの集合，すなわちラベル集合を観測可能なもの $\mathcal{O}$ と不能なもの $\mathcal{O}^c$ に分けることを考える．この場合，観測可能なラベルは $\overline{P}$，$\overline{Q}$ 双方で共通して用いられるもの，すなわち $\mathcal{O} \subseteq \mathcal{L}(\overline{P}) \cap \mathcal{L}(\overline{Q})$ である必要がある．

多重 Ambient Calculus の全体式 $\overline{P}$ に対して，$\overline{P}$ から到達可能なプロセス系の集合を $\mathcal{R}(\overline{P})$ とする．

Ambient Calculus に関する等価性理論の研究として，双模倣等価に基づく研究 [12], [13] や，名前の観測可能性に基づく研究 [14], [15] などが報告されている．しかしこれらの研究の中で報告されている双模倣等価関係の定義は，アンビアントの移動によって生じる合同性の問題に対応するため，文献 [16] などで述べられている一般的な双模倣等価関係の定義に比べ複雑なものとなっている．これに対し，本論文では，全体式間の等価性を議論の対象とする．全体式に対する演算は個別式の追加に限定され，演算子によるプロセス式の合成は行わないため，合同性の問題も限定的なものとなる．そのため本論文で扱う双模倣等価関係は，文献 [16] と同様の一般的な形で定義することができる．

定義 4.1（弱双模倣等価）
2 つの多重 Ambient Calculus の全体式 $\overline{P}$，$\overline{Q}$ と観測可能イベント（遷移ラベル）の集合 $\mathcal{O} \subseteq \mathcal{L}(\overline{P}) \cap \mathcal{L}(\overline{Q})$ に対して，関係 $B \subseteq \mathcal{R}(\overline{P}) \times \mathcal{R}(\overline{Q})$ が以下の性質を満たすとき，関係 B は観測可能イベント集合 $\mathcal{O}$ に対して弱双模倣であるという．また，そのような B が存在するとき，$\overline{P}$ と $\overline{Q}$ は観測可能イベント集合 $\mathcal{O}$ に対して弱双模倣等価であるといい，$\overline{P} \simeq_{\mathcal{O}} \overline{Q}$ と書く．
$(\overline{R}, \overline{S}) \in B$ であるとき以下が成り立つ．ただし $\mathcal{O}^c$ は $\mathcal{O}$ の補集合を表すものとする．
(1) $\overline{R} \rightarrow_l \overline{T} (l \in \mathcal{O})$ ならば $\overline{S} \stackrel{*}{\rightarrow}_{l'} \overline{U}$ $(l' \in (\mathcal{O}^c)^* l (\mathcal{O}^c)^*)$ かつ $(\overline{T}, \overline{U}) \in B$ なる $\overline{U}$ が存在する．
(2) $\overline{R} \rightarrow_l \overline{T} (l \in \mathcal{O}^c)$ ならば $\overline{S} \stackrel{*}{\rightarrow}_{l'} \overline{U}$ $(l' \in (\mathcal{O}^c)^*)$ かつ $(\overline{T}, \overline{U}) \in B$ なる $\overline{U}$ が存在する．
(3) $\overline{S} \rightarrow_l \overline{U} (l \in \mathcal{O})$ ならば $\overline{R} \stackrel{*}{\rightarrow}_{l'} \overline{T}$ $(l' \in (\mathcal{O}^c)^* l (\mathcal{O}^c)^*)$ かつ $(\overline{T}, \overline{U}) \in B$ なる $\overline{T}$ が存在する．
(4) $\overline{S} \rightarrow_l \overline{U} (l \in \mathcal{O}^c)$ ならば $\overline{R} \stackrel{*}{\rightarrow}_{l'} \overline{T} (l' \in (\mathcal{O}^c)^*)$ かつ $(\overline{T}, \overline{U}) \in B$ なる $\overline{T}$ が存在する． □

$\mathcal{O}$ を省略して $\overline{P} \simeq \overline{Q}$ と書くとき，$\mathcal{O} = \mathcal{L}(\overline{P}) \cap \mathcal{L}(\overline{Q})$ であるとする．

$\overline{P} \simeq_{\mathcal{O}} \overline{Q}$ ならば，$\mathcal{O}$ のみが観測可能であるとしたときの $\overline{P}$ から実行可能な観測可能イベント系列の集合と，$\overline{Q}$ から実行可能な観測可能イベント系列の集合が一致する．

4.2 共通する個別式を持つ全体式間の弱双模倣等価性

比較の対象とする全体式を，共通する個別式を持つ，$\overline{P} = (P_1, \ldots, P_l, P_{l+1}, \ldots P_m)$ と $\overline{Q} = (P_1, \ldots, P_l, Q_{l+1}, \ldots Q_n)$ のような形であり，さらにすべての大域アンビアントはそれらの共通個別式 $P_1, \ldots, P_l$ のいずれかに現れる，すなわち $\overline{P}$，$\overline{Q}$ の両方で $A_G = \bigcup_{i=1}^{l} A_G^i$ が成り立つものに限定し，それらに関するイベントはすべて観測可能，すなわち $\mathcal{O} \supseteq \mathcal{L}_G(\overline{P}) = \mathcal{L}_G(\overline{Q})$ であるとする．

また，一方の全体式のみに含まれる個別式 $P_{l+1}, \ldots P_m$，$Q_{l+1}, \ldots Q_n$ で用いられる個別名は重なりを持たない（必要に応じて名前の付け替えを行う），すなわち $A_I^i(\overline{P}) \cap A_I^j(\overline{Q}) = \emptyset$ $(i, j > l)$ であるとする．

そして，特に断らない限り $\overline{P} \simeq_{\mathcal{O}} \overline{Q}$ であると仮定するとき，

$$B = \{((R_1, \ldots, R_m), (S_1, \ldots, S_n)) \mid R_i = S_i (1 \leq i \leq l)\}$$

を $\mathcal{R}(\overline{P})$ と $\mathcal{R}(\overline{Q})$ 間の $\mathcal{O}$ に対する弱双模倣関係であるとする．ただし，l は $\overline{P}$，$\overline{Q}$ の共通個別式の数である．

例 4.2　共通個別式を持つ全体式を比較する例として，同じ船で別の貨物を運搬する次の 2 つの物流の例を考える．

```
P = (Invoice(m, TK, MJ), SHIProute)
Q = (Invoice(n, KB, MJ), SHIProute)
```

この場合 P と Q の共通個別式は SHIProute であり，すべての大域アンビアントはその式に現れるので，上記の制限を満たしている．詳細は省略するが，これらの式は観測可能イベント集合 $\mathcal{O} = \mathcal{L}(\overline{P}) \cap \mathcal{L}(\overline{Q})$ に対して弱双模倣等価であり，このことは両者に共通して現れるアンビアントである SHIP の，共通して現れるアンビアント TK，KB，MJ への出入りということに関して両者は同じ振舞いをすることも表している． □

以降では繁雑さを避けるため，$i \leq j$ であるような全体式中の「$X_i, \ldots, X_j$」を $X_{i \ldots j}$ と略記することにする．

共通個別式を持つ 2 つの全体式の特別な場合として，$\overline{P} = (P_{1 \ldots m})$，$\overline{Q} = (P_{1 \ldots m}, Q_{m+1 \ldots n})$ のような場合，すなわち一方が他方にいくつかの個別式を追加したものであるような場合を考えることができる．そのような $\overline{P}$，$\overline{Q}$ に対して $\overline{P} \simeq_{\mathcal{L}(\overline{P})} \overline{Q}$ が成り立つならば，$\overline{P}$ において，あるイベント系列が実行可能であるならば，$\overline{Q}$ でも，観測不能な

遷移を適宜挿入することで同じイベント系列が実行可能である．これは $\overline{Q}$ で追加された個別式 $Q_{m+1},\ldots,Q_n$ がもとの $\overline{P}$ の振舞いを阻害していないと解釈することができる．

例 4.3　例 4.2 の $\overline{\mathtt{P}}$，$\overline{\mathtt{Q}}$ は貨物の取扱いを含まない式 $\overline{\mathtt{R}}$=(SHIProute) とも $\mathcal{O} = \mathcal{L}(\overline{P}) \cap \mathcal{L}(\overline{Q}) = \mathcal{L}(\overline{R})$ に対して弱双模倣等価であることが確認できる．これは Invoice(m, TK, MJ)，Invoice(n, KB, MJ) で記述した貨物の取扱いが正常に完了するならば，SHIProute で記述された SHIP の運航を阻害しないことを表している．　□

一般的な合同性に代えて，全体式間の弱双模倣等価性が，一定の条件を満たす式の追加について閉じていることを表す以下の性質が成り立つ．

定理 4.4　$\overline{P} = (P_{0\ldots m})$，$\overline{Q} = (P_{0\ldots l}, Q_{l+1\ldots n})$ とし，それらから P_0 を除去した全体式 $\overline{P^-} = (P_{1\ldots m})$，$\overline{Q^-} = (P_{1\ldots l}, Q_{l+1\ldots n})$ とし，それらの大域遷移ラベルの集合は同じとする．$\mathcal{L}_G(\overline{P}) \subseteq \mathcal{O} \subseteq \mathcal{L}(\overline{P^-}) \cap \mathcal{L}(\overline{Q^-})$ なる観測可能イベント集合 $\mathcal{O}$ について，$\overline{P^-} \simeq_{\mathcal{O}} \overline{Q^-}$ ならば $\overline{P} \simeq_{\mathcal{O}\cup\mathcal{L}_I^0} \overline{Q}$ が成り立つ．

[証明] $\overline{P}$，$\overline{Q}$，$\overline{P^-}$，$\overline{Q^-}$ の大域遷移のラベル集合が同じとしているため P_0 は他の個別式に現れない大域遷移のラベルを持たないので，$\overline{P}$，$\overline{Q}$ に現れるラベル集合は $\mathcal{O}$，$\mathcal{O}^c = \mathcal{L}(\overline{P^-}) \cup \mathcal{L}(\overline{R^-}) - \mathcal{O}$，$\mathcal{L}_I^0$ に分割できる．また $\mathcal{L}_G(\overline{P}) \subseteq \mathcal{O}$ なので $\mathcal{L}^0 \cap \mathcal{O}^c = \emptyset$ である．

$B = \{((R_{0\ldots m}), (S_{0\ldots n})) \mid R_i = S_i (0 \le i \le l)\}$ とし，これが弱双模倣関係であることを示す．$(\overline{R}, \overline{S}) \in B$ なる $\overline{R} = (R_{0\ldots m})$, $\overline{S} = (R_{0\ldots l}, S_{l+1\ldots n})$ を考える．このとき，補題 3.6 より，$\overline{R^-} = (R_{1\ldots m}) \in \mathcal{R}(\overline{P^-})$，$\overline{S^-} = (R_{1\ldots l}, S_{l+1\ldots n}) \in \mathcal{R}(\overline{Q^-})$ が成立つ．$B^- = \{((R_{1\ldots m}), (S_{1\ldots n})) \mid R_i = S_i (0 \le i \le l)\}$ を $\overline{P^-}$ と $\overline{Q^-}$ が弱双模倣等価であることを保証する弱双模倣関係とする．$(\overline{R^-}, \overline{S^-}) \in B^-$ であり，B^- が弱双模倣関係であることより，$(\overline{R^-}, \overline{S^-})$ について定義 4.1 の (1)〜(4) が成り立つ．以下で，$(\overline{R}, \overline{S})$ について定義 4.1 の (1)〜(4) の成立を確認する．

(1-1) $l \in \mathcal{O}$ の場合，$\overline{R} \to_l \overline{T} = (T_{0\ldots m})$ とすると，補題 3.5 より $R_0 \to_l T_0$ および $\overline{R^-} \to_l \overline{T^-} = (T_{1\ldots m})$，が成り立つ．$(\overline{R^-}, \overline{S^-})$ について定義 4.1 の (1) が成り立つことより，$\overline{S^-} \stackrel{*}{\to}_{l'} \overline{U^-} = (T_{1\ldots l}, U_{l+1\ldots n})$，$l' \in (\mathcal{O}^c)^* l (\mathcal{O}^c)^*$ なる $\overline{U^-}$ が存在する．一方，$\mathcal{L}^0 \cap \mathcal{O}^c = \emptyset$ と $R_0 \to_l T_0$ より $R_0 \stackrel{*}{\to}_{l'} T_0$ なので，結局 $\overline{S} \stackrel{*}{\to}_{l'} \overline{U} = (T_{0\ldots l}, U_{l+1\ldots n})$ が導かれ，$(\overline{T}, \overline{U}) \in B$ である．

(1-2) $l \in \mathcal{L}_I^0$ とする．l が個別遷移なので，$1 \le i \le m, n$ に対して $R_i \to_l R_i$, $S_i \to_l S_i$．このため $\overline{R} \to_l \overline{T} = (T_0, R_{1\ldots m})$ ならば $\overline{S} \to_l \overline{U} = (T_0, R_{1\ldots l}, S_{l+1\ldots n})$ となり，$(\overline{T}, \overline{U}) \in B$.

(2) $l \in \mathcal{O}^c$ とする．$\mathcal{L}^0 \cap \mathcal{O}^c = \emptyset$ より $R_0 \to_l R_0$．また $(\overline{R^-}, \overline{S^-})$ について定義 4.1 (2) が成り立つことより，$\overline{R} \to_l \overline{T} = (R_0, T_{1\ldots m})$ ならば $\overline{S} \to_l \overline{U} = (R_0, T_{1\ldots l}, U_{l+1\ldots n})$ となり，$(\overline{T}, \overline{U}) \in B$.

(3)，(4) については (1)，(2) と同様である．　□

たとえば，例 4.2 で取り上げた 2 つの式が弱双模倣等価であることより，それらの式に新たな貨物の取扱いを記述した式を追加した際，追加した式が $\overline{\mathtt{P}}$ の振舞いを阻害しないときかつそのときのみ $\overline{\mathtt{Q}}$ の振舞いを阻害しないことが保証される．

5. 議論

ここでは多重 Ambient Calculus の全体式として複数の個別式を用いて物流を記述した場合と，通常の Ambient Calculus の式として単一の式で物流システムを記述した場合の記述性について比較する．

単純な物流の例として 2 つの貨物 c1，c2 を東京港（TK）から神戸港（KB）へ，東京港に入港後，神戸港，門司港（MJ）に回航する船 SHIP で運搬する場合を考える．

多重 Ambient Calculus で記述する場合，例 3.7 と同様に以下のように記述できる．

```
(Invoice(c1, TK, KB),
 Invoice(c2, TK, KB),
 SHIProute)
```

一方，Ambient Calculus の単一式で記述する場合，Invoice(c1, TK, KB) 同様神戸港での荷降ろしを省略した場合，たとえば文献 [8] では以下のようなスタイルで記述することを考えている．

```
  SHIP[ in TK. ( load[out SHIP. in CY. in c1]
               | load[out SHIP. in CY. in c2]
               | open lcomp1. open lcomp2.
                      out TK. in KB. out KB.
                      in MJ. out MJ)]
| TK[ CY[ c1[open load. out CY. in SHIP.
             lcomp1[out c1]]
       | c2[open load. out CY. in SHIP.
             lcomp2[out c2]]]
| KB[]
| MJ[]
```

この場合，lcomp1 に関する記述と lcomp2 に関する記述が非対称になっており，c1 の積み込み確認「open lcomp1」，c2 の積み込み確認「open lcomp2」を実際に積み込まれた順序とは関係なく必ず c1，c2 の順に行うというような記述になっており，記述しようとする物流計画が本来持っている対称性，並行性を十分表現しているとはいえない．文献 [9] では，式の対称性を利用してモデル検査の際の探索空間を縮小する方法を議論しており，たとえば，上記の式は lcomp1，lcomp2 をともに lcomp とすることで，対称性を持つ式と見なせ，モデル検査の効率化が図れるとしている．しかしながらそのような記述スタイルでは，積み込まれた貨物を個別に確認するのではなく，単に数を確認しているだけになってしまい，貨物の取り違えを見逃すことになりかねない．

以上のように，多重 Ambient Calculus により物流システムを記述することにより，通常の Ambient Calculus で記述した場合と比較して，物流計画の持つ対称性，並行性をより適切かつ容易に表現することができ，記述性，了解性が向上したと考えられる．さらに，物流計画の持つ対称性をより直接的に表現する形になっているため，文献 [9] でのモデル検査法を応用した効率的なモデル検査が可能になると期待される．

また，これらの記述に新たな貨物の輸送を追加，あるいは貨物の輸送を削除する場合について考えると，多重 Ambient Calculus では 3.3 節で見たように，その貨物の送り状に相当する個別式を追加もしくは削除すればよいのに対し，単一式で記述した場合，式中の複数の箇所に新たな部分式を埋め込んだり削除したりする必要が生じ，式の取扱いの柔軟性の観点からも多重 Ambient Calculus が有用であるといえる．

一方，多重 Ambient Calculus による記述では大域アンビアントに関する同様の記述を複数の個別式に重複して記述する必要があるので，全体式が冗長なものになってしまうという問題がある．これについては何らかの圧縮法が必要になると考えられる．

6. おわりに

本論文では多数の貨物を取り扱う物流システムを記述するという観点から，従来の Ambient Calculus を拡張した多重 Ambient Calculus を提案した．今後，多重 Ambient Calculus で記述されたプロセス式に基づく物流検査システムを構築するとともに，多重 Ambient Calculus で記述された物流システムの正当性を確認するためのモデル検査の効率化について検討する予定である．

謝辞　本研究は科研費（22500040）の助成を受けたものである．

参考文献

[1] Cardelli, L. and Gordon, A.D.: Mobile Ambients, *Theoretical Computer Science*, Vol.240, pp.177–213 (2000).

[2] 吉岡信和，田原康之，本位田真一：モバイルエージェントによる柔軟なコンテンツ流通を実現するアクティブコンテンツ，情報処理学会論文誌：データベース，Vol.44, No.SIG18(TOD 20), pp.45–57 (2003).

[3] Giannini, P., Sangiorgi, D. and Valente, A.: Safe Ambients: Abstract Machine and Distributed Implementation, *Science of Computer Programming*, Vol.59, Issue 3, pp.209–249 (2006).

[4] 岡田翔太, 馬谷誠二, 林　奉行, 八杉昌宏, 湯淺太一：Safe Ambients のための Java フレームワーク，情報処理学会論文誌　プログラミング，Vol.4, No.3, pp.26–41 (2011).

[5] 日本経済新聞社：海上コンテナの位置追跡，日本経済新聞 2007 年 1 月 24 日発行夕刊 3 面 (2007).

[6] 国土交通省：「メコン地域陸路実用化走行試験」—インドシナ半島物流を変える陸路物流の実用化へのチャレンジ，入手先 ⟨http://www.mlit.go.jp/kisha/kisha07/15/151018_.html⟩.

[7] 国土交通省：米国国土安全保障省及び国土交通省による海上貨物追跡タグシステム（MATTS）の通信能力実証実験，入手先 ⟨http://www.mlit.go.jp/kisha/kisha07/11/110427_.html⟩.

[8] 森本大輔，加藤　暢，樋口昌宏：Ambient Calculus を用いた物流検査システム，情報処理学会論文誌：プログラミング，Vol.48, No.SIG10(PRO33), pp.151–164 (2007).

[9] 加藤　暢，樋口昌宏，植田直人：物流システムに対する Ambient Logic モデル検査システム，情報処理学会論文誌　数理モデル化と応用，Vol.3, No.1, pp.73–86 (2010).

[10] 片山立志：よくわかる貿易書類入門，日本能率協会マネジメントセンター (2006).

[11] 商船三井ロジスティック：インボイス・パッキングリスト製作支援，入手先 ⟨http://www.mol-logistics.co.jp/japan/jp/support/invoice/sample_invoice.shtml⟩.

[12] Merro, M. and Hennessy, M.: Bisimulation Congruences in Safe Ambients, *POPL '02, Proc. 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp.71–80, ACM (2002).

[13] Merro, M. and Hennessy, M.: A Bisimulation-based Semantic Theory of Safe Ambients, *Trans. Programming Languages and Systems*, Vol.28, pp.290–330 (2006).

[14] Gordon, A.D. and Cardelli, L.: Equational Properties of Mobile Ambients, *Mathematical Structures in Computer Science*, Vol.13, No.3, pp.371–408 (2003).

[15] Kato, T.: An Equational Relation for The Typed Ambient Calculus, *IPSJ Trans. Programming*, Vol.48, No.SIG10(PRO33), pp.90–100 (2007).

[16] Parrow, J.: An Introduction to the π-Calculus, *HANDBOOK OF PROCESS ALGEBRA*, Bergstra, J.A., Ponse, A. and Smolka, S.A., Eds., NORTH-HOLLAND (2001).

樋口 昌宏　（正会員）

1983 年大阪大学基礎工学部情報工学科卒業．1985 年同大学大学院博士前期課程修了．（株）富士通研究所研究員，大阪大学講師等を経て，現在，近畿大学理工学部情報学科准教授．博士（工学）．分散システムの記述，検証，試験に関する研究に従事．

加藤 暢　（正会員）

1997 年岡山大学大学院自然科学研究科博士課程修了．博士（工学）．1998 年日本学術振興会特別研究員（PD）．2000 年より近畿大学理工学部講師．現在，准教授．並行論理型言語の意味論，プロセス代数の研究に従事．