情報処理学会論文誌　プログラミング　Vol.6　No.2　1–12 (Aug. 2013)

多重Ambient CalculusとUHF帯RFID機器を用いた海上物流監視システム

橋本 隆弘[1]　加藤 暢[2,a)]　樋口 昌宏[2]

受付日 2012年11月12日, 採録日 2013年4月24日

概要：我々は，多重 Ambient Calculus（MAC）による物流システムの記述方法の研究および，実際の物流が正しく行われているかを監視するシステムの開発を行っている．物流システムは，より小さなパッケージをより大きなパッケージに収容する階層構造を持っている．これに対し，プロセス代数の一種である MAC は，階層構造を持ち動的に構造が変化するシステムを形式的に記述できる．MAC は，通常の Ambient Calculus（AC）と異なり，個々の貨物の取扱いを個別のプロセス式として記述し，それらの集合により物流システム全体をモデル化するものである．この特徴により，通常の AC 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，貨物が追加，削除される際のプロセス式の変更も容易となる．本稿では，MAC と UHF 帯 RFID 機器を用いた物流監視システムを提案する．本システムは，実際のコンテナ輸送で使われる数種類の書類をもとに，各コンテナの搬送経路を表現する MAC 式を自動生成するシステム，MAC の個別式を適切な端末へと分散するシステム，および RFID を用いて検知した貨物の動きや，船の乗り換えが MAC 式の遷移に沿ったものであるかを監視するシステムからなる．また，本システムの有用性を示すために，車をコンテナと見立てて行った屋外実験についても述べる．

キーワード：Ambient Calculus，RFID，海上物流

A Handling Management System for Freight with the Multiple Ambient Calculus and UHF RFID Tags

Takahiro Hashimoto[1]　Toru Kato[2,a)]　Masahiro Higuchi[2]

Received: November 12, 2012, Accepted: April 24, 2013

Abstract: We are investigating a way of modeling freight system in the Multiple Ambient Calculus (MAC) and the management systems ensuring the correctness of container handling during shipping using the model. Freight systems have nested structures, i.e., packages (e.g. containers) containing smaller packages (e.g. luggage) that are accommodated by a larger entity such as a container ship. MAC is a formal description language that is suitable for modeling such systems with the nested structures that dynamically change. Using MAC, whole the freight system is modeled by a set of formulae each of which represents the handling of each containers while, using the Ambient Calculus, the freight systems are modeled by a large formula. Thanks to this feature of MAC, we can express symmetric and concurrent property of freight systems more appropriately and adding or retracting formulae (representing adding or canceling containers for a voyage) becomes easier than using AC. The management system propsesed in this paper consists of three sub systems. The first generates formulae that represent freight systems based on several trading documents, the second deploys the formulae to each devices (PCs and RFID tags), the last senses the handling of containers by using UHF RFID devices and checks their consistency with the transitions of corresponding formulae. We also show the results of preliminary experiments in which a car is used as a container.

Keywords: Ambient Calculus, RFID, maritime freight systems

1. はじめに

近年，物流の世界では，貨物の流通量の増加にともないコンテナ管理の重要性が高まっており，コンテナをどのように管理していくか様々な研究が行われている [7], [8]．我々は現在，物流書類を用いて物流システム全体をモデル化し，コンテナや船の動きを自動で監視するシステムに関する研究を行っている．

Ambient Calculus（AC）[1] は，動的な階層構造を代数式で表現することができ，ネットワーク間を移動するようなモバイルプロセスの記述に特化したプロセス代数である．一方海上物流にも，大きな枠組みであるコンテナ船が小さな枠組みであるコンテナを積んでいるという階層構造が存在している．そしてその階層構造は，コンテナ船がある港から別な港へと移動したり，コンテナの積込みや積降ろしをするとき動的に変化する．

我々は AC と物流システムの双方が持つ動的な階層構造という類似点に着目し，文献 [11] において，物流書類の内容を AC のプロセス式を用いて表現し，実際の物流がそのプロセス式の記述のとおりに行われているかを監視するシステムを提案した．しかし通常の AC を用いると物流全体を単一のプロセス式でモデル化するため，プロセス式が複雑になりがちであり，取り扱うコンテナの追加や削除の際に式全体を修正しなければならなかった．そのため，このような実際の物流の動的な変化に対応するには，AC を用いたモデル化では柔軟性に欠けることが明らかになった．そこで我々は文献 [12] において，物流システムの動的な変化に柔軟に対応することのできる多重 Ambient Calculus（MAC）を提案した．MAC では，船やコンテナを個別式と呼ぶ個別のプロセス式で表現し，複数の個別式からなる全体式を物流計画のモデルとする．こうすることで，積み込むコンテナに追加があった場合，式の変更はコンテナに対応する個別式を追加するだけでよいことになる．

本稿では，この MAC と UHF 帯 RFID 機器を利用した物流監視システムを提案する．本システムは，物流書類から MAC のプロセス式を生成するシステム，MAC の式を遷移させることのできる処理系，および RFID 機器を用いて検知した実際のコンテナの取扱いをプロセス式の遷移と対比させ，その取扱いが正しいものであるかどうかを監視するシステムからなる．

2 章では，本研究で対象としている物流システムについて述べる．3 章では本研究で基礎としている AC について，その構文規則，遷移規則を述べる．4 章では，本稿で利用する MAC について述べ，AC のプロセス式を用いた物流のモデル化と対比することで，MAC の利点を具体的に説明する．5 章では，本稿で提案する物流監視システムについて，その構成や機器配置，およびその動作を説明する．6 章では本システムの実現性を示すために行ったいくつかの実験結果について述べる．

[1] シー・エス・イー株式会社
CSE Co.,Ltd., Shibuya, Tokyo 150–0002, Japan
[2] 近畿大学
Kinki University, Higashiosaka, Osaka 577–8502, Japan
[a)] kato@info.kindai.ac.jp

2. FCL 海上物流システム

本章では，本研究で対象とするコンテナを用いた海上物流システムについて説明する．

2.1 FCL 海上物流システムとハブ・アンド・スポーク・システム

コンテナを用いた海上物流システムは，大きく LCL（Less than Container Load）と FCL（Full Container Load）に分かれる．前者は 1 つのコンテナを複数のの荷主が利用する形態，後者は 1 つのコンテナを 1 人の荷主が利用する形態である [14]．本研究ではその構造の単純さから後者を対象とする．

海上物流の分野では近年ハブ・アンド・スポーク・システムと呼ばれる輸送システムが普及している [9]．これは海運会社間の提携関係の下で，各海運会社の拠点港，あるいは中継港をハブ港，ハブ港の周りのローカルな港をフィーダ港とし，ハブ港を中心とした輸送を行うことにより，効率的な輸送を行う輸送システムである．ハブ・アンド・スポーク・システムでは，船の運行計画は天候の影響などにより刻々と変化するため，乗り換えの船は通常中継港にコンテナが届いてから決定される．

2.2 物流定義書類とそれに基づく物流作業

物流システムの動作は書類を使って規定され，その書類に基づいてコンテナの取扱いあるいは船の運航などの作業が行われる．本節では，物流に用いられる書類と，それに基づく作業について説明する．

荷主はコンテナの荷出し港と最終仕向港を指定した送り状 [10] を作成し，それに基づいて輸送業者がコンテナ輸送を行う．コンテナの船への荷積み，港への荷下ろし作業は輸送業者が各港で作成する B/L Instructions に従って行われる．B/L Instructions には，輸送に使用されるコンテナの本数と各コンテナのコンテナ番号のほか，荷積み港，荷降ろし港，コンテナが積み込まれる船といった情報が記載されている．また，輸送業者は各船舶の航路表を作成し，それに従って船の運航が行われる．航路表には船を識別するための船名と航海番号，その船が寄港する港が寄港順に記述された一覧が航路として記載されている．

また，船が出港する際，その港におけるすべての荷降ろし，荷積み作業は対象となる船が港に到着してからしか実行してはならない．このように海上物流では，船の運航とコンテナ取扱い作業の間の同期が必要である．

以上の書類のうち送り状と航路表は，それぞれ荷主，輸送業者によって任意に作成されるものである．一方，B/L Instructions は輸送業者が送り状の記載に従って，コンテナが最終仕向港に届くように各港で作成される．B/L Instructions の作成時には，そのコンテナを次にどの船に乗せてどの港まで運ぶかを輸送業者が選択する．この選択が適切に行われることによって，効率的な物流が可能となる．そこで本稿では，上記の書類に加えて，各港において，コンテナの最終仕向港ごとに最適な船と荷降ろし港を指定した輸送航路指定表が作成されているものとする．

3. Ambient Calculus

Ambient Calculus 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表現することができる．この特徴を活かして，ネットワーク上で動的に情報をやりとりするための，様々な実装が提案されている [4], [6]．また，この特徴により物流システムの持つ階層構造を簡潔に表現できる．

3.1 Ambient Calculus の構文規則と遷移規則

構文規則，遷移規則は文献 [1] の定義をそのまま利用する．

定義 3.1（構文規則）

$P, Q ::=$	*processes*	$M ::=$	*capabilities*
$(\nu n)P$	restriction	x	variable
0	inactivity	n	name
$P\|Q$	composition	*in* M	can enter into M
$!P$	replication	*out* M	can exit out of M
$M[P]$	ambient	*open* M	can open M
$M.P$	action	ϵ	null
$(x).P$	input action	$M.M'$	path
$\langle M \rangle$	async action		

定義 3.2（遷移規則）

$n[\ in\ m.P \mid Q\ ] \mid m[\ R\ ] \longrightarrow m[\ n[\ P \mid Q\ ] \mid R\ ]$

$m[\ n[\ out\ m.P \mid Q\ ] \mid R\ ] \longrightarrow n[\ P \mid Q\ ] \mid m[\ R\ ]$

$open\ n.P \mid n[\ Q\ ] \longrightarrow P \mid Q$

$(\ x\ ).P \mid \langle M \rangle \longrightarrow P\{\ x \leftarrow M\ \}$

定義 3.1，3.2 の直観的な説明は文献 [12] に譲る．

3.2 Ambient calculus を用いた物流計画の記述例

東京港（TK）から神戸港（KB）へ航行する船 SHIP を用い，2 個のコンテナ co1，co2 を東京港のコンテナヤード（CY）で積み込んで，神戸港へ輸送する物流は，以下の式により表現できる．

例 3.1（記述例）

```
SEA[SHIP[in TK.open lcomp.open lcomp.
    out TK.in KB.open ulcomp
    .open ulcomp]
|TK[load[checkin(SHIP).in CY]
     |CY[co1[checkout(load) CY
       .in SHIP.lcomp[out co1]
      |checkout(uload)SHIP.in CY
      .ulcomp[out co1.out CY.in SHIP]]
    |co2[checkout(load)CY.
      in SHIP.lcomp[out co2]
      | checkout(uload)SHIP.in CY
       .ulcomp[out co2.out CY.in SHIP]]]]
 |KB[uload[in SHIP]
     |CY[ ]]
```

この式において，コンテナ ambient が勝手に動かないように下記のような同期をとらなければならないため，TK が持つ load のような制御のための ambient が必要になる．

- 船が港に入る
- コンテナはコンテナヤードの外に出る
- コンテナが船に乗る

また，checkin および checkout は物流監視システムのために作られた，同期の制御のための動作である．これらの同期をとるための記述が，co1，co2 のコンテナにそれぞれ収まればいいのだが，式中にばらばらに存在してしまう．そのため，AC を用いて物流システムを記述した場合，コンテナを追加したり削除したりする際の式更改の手順が非常に複雑になってしまう．

4. 多重 Ambient Calculus（MAC）

本研究では，MAC を用い，プロセス式の組により物流システムを記述する．MAC 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 ambient 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 MAC の遷移規則に導入することで，プロセス間の同期を表す．MAC では n 個のプロセス式の組 $\overline{P} = (P_1, ..., P_n)$ で 1 つの物流計画全体を表現する．個々のプロセス式 P_i を個別式と呼び，$\overline{P}$ を全体式と呼ぶ．また，共通名を持つ ambient を大域 ambient と呼ぶ．大域 ambient は複数の個別式に共通に現れ，大域 ambient に関する遷移はそれらを共有する個別式で同時に発生する．これにより，個別式間での同期を表現できるので，全体式で物流計画を表現できる．

したがって，MAC 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 つのコンテナに関する部分のみを記述するということが可能となる．

4.1　補助定義

ここで，MAC の遷移規則の定義のために文献 [12] で導入されたいくつかの補助的な AC の定義について説明しておく．

遷移規則の定義 3.2 では ambient 構文「$n[P]$」を用いて記述された ambient であっても，action 構文「$M.P$」を用いて何らかの capability に先行されているもの，およびその子孫の ambient は遷移に関与しない．そのような ambient を非活性といい，そうでないものを活性と呼ぶ．プロセス式 P から活性 ambient 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 4.1（ambient 階層の抽出 [12]）

$\mathcal{H}((\nu n)P) = (\nu n)(\mathcal{H}(P))$	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

プロセス式の遷移により ambient 階層は変化する．ambient 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 4.2（遷移ラベル [12]）

$n[in\ m.P|Q]|m[R] \rightarrow_l m[n[P|Q]|R],\ l =$ “n enter m”
$m[n[out\ m.P|Q]|R] \rightarrow_l n[P|Q]|m[R],\ l =$ “n exit m”
$open\ n.P|n[Q] \rightarrow_l P|Q,\ l =$ “n disappear”

4.2　大域 ambient

MAC では，ambient 名の名前空間を，個々の個別式のみに関連する個別 ambient 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 ambient のためのものに分割する．全体式 $\overline{P}$ の大域 ambient 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 ambient の集合を $A^i(\overline{P})$，大域 ambient 名の集合を $A^i_G(\overline{P})$，個別 ambient 名の集合を $A^i_I(\overline{P})$ と書く．以降，大域 ambient 名は先頭に大文字を用いて区別する．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 ambient 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．

個別式の ambient 階層 S から，指定した ambient 集合 X を持つ物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 4.3（Ambient 名射影 [12]）

$\mathcal{G}_X((\nu n)P) = (\nu n)\mathcal{G}_X(P)$	restriction
$\mathcal{G}_X(0) = 0$	inactivity
$\mathcal{G}_X(P\|Q) = \mathcal{G}_X(P)\|\mathcal{G}_X(Q)$	composition
$\mathcal{G}_X(n[P]) = n[\mathcal{G}_X(P)], \text{if } n \in X$	glob.ambient
$\mathcal{G}_X(n[P]) = \mathcal{G}_X(P), \text{if } n \notin X$	indi.ambient

定義 4.4（大域 ambient 階層 [12]）

全体式 $\overline{P} = (P_1, ..., P_n)$ 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 ambient の ambient 階層 H が存在するとき，これを $\overline{P}$ の大域 ambient 階層といい，その集合を $\mathcal{H}_G(\overline{P})$ で表記する．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 ambient 階層である．一方，

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

では，$\mathcal{G}_{A_G}(\mathcal{H}(P_i)) = \mathcal{G}_{A^i}(H)$ 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 ambient 階層は一意ではない．以降，プロセス式 $\overline{P}$ に個別式 Q を追加した式を $(\overline{P}, Q)$ と書く．

4.3　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 ambient 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 4.5（全体式の遷移規則 [12]）

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

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

4.4　記述例

MAC を用いた物流システムの記述例を示す．東京港 (TK) から神戸港 (KB) 行きのコンテナ co1 を，東京港のコンテナヤード (CY) から積み込んで神戸港へ航行する船 SHIP で輸送する物流は式 (1) により表現できる．

```
(TK[CY[co1[open load.out CY
 .in SHIP.lcomp[out co1]
|open uload.out SHIP.in CY]]]
|SHIP[in TK.(load[out SHIP.in CY.in co1]
|open lcomp.out TK.in KB.uload[in co1])]
|KB[CY[]],
TK[]|KB[]|SHIP[in TK.out TK.in KB]).
```
(1)

式 (1) を見てみると，第 1 成分と第 2 成分で SHIP enter TK が実行可能である．

ここで第 2 成分のみを見れば，SHIP exit TK が実行可能であるが，第 1 成分では SHIP ambient の out TK は open lcomp に先行されており，この遷移は実行可能ではない．したがって第 1 成分の個別遷移のみ実行可能であり，式 (2) のように遷移する．

```
(TK[SHIP[out TK.in KB.uload[in co1]|
       co1[open uload.out SHIP.in CY]]
   |CY[]]
  |KB[CY[]],
  TK[SHIP[out TK.in KB]]|KB[]).
```
(2)

式 (2) では大域遷移 SHIP exit TK が可能になり，さらに SHIP enter KB が実行可能であるため式 (3) のように遷移する．

```
(TK[CY[]]
|KB[SHIP[uload[in co1]
  |co1[open uload.out SHIP.in CY]]|CY[]],
TK[]|KB[SHIP[]])
```
(3)

その後コンテナが神戸港のコンテナヤードへと移動する動作を表す co1 に関する遷移が行われ，式 (4) となる．

```
(TK[CY[]]|KB[SHIP[]|CY[co1[]]],
TK[]|KB[SHIP[]]).
```
(4)

第 1 成分は貿易書類から得られる式であり，第 2 成分は航路表から得られる式である．第 1 成分は荷受港と仕向港のみが指定されており，実際の航路は第 2 成分によって指定されている．また，同じ船を用いて別のコンテナを輸送する場合は，それらのコンテナに関して第 1 成分のように式を記述し，全体式に追加すればよい．

このように．MAC を用いて複数の式の組によって物流計画を記述すると，たとえば輸送するコンテナに追加があっても，コンテナに関する個別式を追加するだけでよい．

たとえば，例 3.1 のようにコンテナを 2 つ運びたいのであれば，式 (1) の第 1 成分の co1 を co2 に変えた式を，新たに式 (1) の中に追加すればよい．

通常の Ambient Calculus を用いて記述する場合と比較して，例 3.1 では東京港から神戸港に運ぶという式のためある程度簡潔ではあるが，たとえば大阪港から神戸港へ，神戸港から福岡港へ，といったようなコンテナの式があったとすれば，それだけコンテナに関する記述が式全体の中に散乱してしまう．一方 MAC を用いる場合，そのコンテナはどの船に乗ってどの港へ行くのか，という個別式を書けばよい．しかし，通常の AC と比べ，コンテナの数だけ港 ambient や船 ambient は存在しなければならないため，ambient や capability の数は増加することになる．AC でコンテナ数が 1 個の場合の遷移式で数えたものも含め，まとめたものが**表 1** である．AC よりも MAC のほうが，コンテナ数が増えたときの ambient 数, capability 数ともに増加量が多いのが分かる．このプロセス式の場合，コンテナ数を n とすると AC の ambient 数は $3n+8$，capability 数は $9n+8$ であり，MAC では ambient 数は $9n+3$，capability 数は $15n+3$ となるため，コンテナ数が増えるにつれてその差は非常に大きくなる．

しかし，MAC を用いることにより，コンテナの追加や削除の際に，部分式のまとまりを追加，削除すればよいということになる．たとえば東京港，名古屋港，大阪港，神戸港に関連する物流の監視を行っている途中に，新たに東京港から名古屋港まで運ぶコンテナの追加を行う場合，AC では式全体の再生成をし，監視を行うために各港の PC に式を渡さなければならなくなる．一方 MAC では，その部分式のみを生成し，全体式と組み合わせるだけでよく，式を渡すのも東京港の PC だけでよい．このように，MAC を用いることでコンテナの追加に対しても監視システムの実装が簡潔になる．この利点は，対象とする物流の規模が大きくなるほど顕著になる．

また，物流管理システムの信頼性の向上が重要となる．我々は AC の記述に基づいたモデル検査システムを提案した [5]．モデル検査システムでは，多数のコンテナを扱うことによるモデル検査の際の状態空間爆発を緩和するた

表 1　AC と MAC の比較
Table 1　Comparison between AC and MAC.

	ambient 数	capability 数
AC コンテナ 1 個	11	17
AC コンテナ 2 個	14	26
MAC コンテナ 1 個	12	18
MAC コンテナ 2 個	21	33

め，プロセス式の対称性の検出と対称性を利用した partial order reduction [2] を行っている．しかしそのような方法を用いたとしても検査可能な物流計画はコンテナ数 200 個程度が限界であるため，さらなる効率化が必要であった．

この問題に対し我々は，MAC を用いて物流計画を表現することで，数千個のコンテナを扱う大規模なプロセス式に対するモデル検査を，その式と弱双模倣等価性なより小規模なプロセス式のモデル検査に帰着させることが可能になることを示した [13]．このモデル検査をコンテナの輸送開始前に行うことで，所期の性質を満たすことが確認されたプロセス式を用いて，正確な物流監視を行うことが可能となる．

5. 多重 Ambient Calculus を用いた物流監視システム

本研究で提案する物流監視システムの目的は以下のとおりである．

- 荷物を収容したコンテナや，コンテナを積んだコンテナ船が輸送計画どおり正しく移動することを確かめる．
- 正しくない移動を行った場合に通知する．
- 現在のコンテナやコンテナ船の位置を把握する．

以上の目的を達成するために以下の実装を行った．

- 物流書類からプロセス式を自動生成するシステム
- 物を表現する ambient ごとの ambient 式の分散処理
- 物の動きとプロセス式遷移との照合
- 上記関連付けにともなう動作エラーの表示

本システムは，コンテナの移動が物流書類の内容を記述した ambient 式の遷移に沿ったものであるかを調べるものである．そのために本システムでは，RFID を用いてコンテナの移動を検知するよう実装した．

5.1 機器の配置と分散処理について

本システムで用いる PC は以下のとおりである．

- ゲートをコンテナが通りコンテナヤードへと入ることを監視するためのゲート PC（5.3 節）
- コンテナヤードから船へのコンテナの積み込み，および船からコンテナヤードへの積み降ろしを監視するためにクレーンに取り付ける PC（5.4 節）
- 積み込むべきコンテナが積まれてから，あるいは積み降ろすべきコンテナが積み降ろされてから船が出航するといった同期的な動作を監視するための船 PC（5.8 節）

この中でクレーン PC には MAC の遷移を行うための処理系を 2 つ持たせる．1 つはコンテナヤードに関する式を，もう 1 つは船に関する式の処理を担当する．これらの処理系は，渡された式を組み合わせ，遷移が行えるかどうかを確認する．それによって正しいコンテナの移動であるかを監視する．

またこれらとは別に，本システムではコンテナに貼り付けた RFID タグに自動的に MAC のプロセス式を書き込む処理を行うが，そのためにはプロセス式と RFID タグの ID を管理するためのデータベースを持つ PC が必要となる．これについては 5.3 節で詳述する．さらに，これらの PC を管理するトップサーバ的な役割を持つ PC を使用する．大まかな機器の配置を**図 1** に示す．

本システムでは，1 台の PC で監視を行う集中型ではなく，上記のような複数の PC を用いる分散型でシステムを構築している．実際の物（コンテナやコンテナ船など）が移動すると，その物を表現する ambient が現実の移動と同じように遷移可能であるかどうかの検査が，プロセス式を基に行われるが，その際に通信が発生する．物流システム全体を表現する式を 1 つのサーバで管理していた場合，多くのコンテナ移動に関する通信が 1 つのサーバに集中する．分散処理を行うことにより，このような通信の集中をさけることができる．

また，1 つのサーバで集中管理していた場合，気象状態など何らかの理由でサーバとの通信ができなくなる可能性がある．本システムの対象は，屋外でのコンテナ積み込み積み降しであり，遠方の PC とつねに通信可能であるとは限らない．各コンテナに取り付けたタグに情報を持たせることで，コンテナが船や港と通信できない状態でもタグ内の情報だけで監視作業を問題なく続けることができる．

5.2 本システム内での通信

本システムで用いる PC 間では，プロセス式やタグ ID などの文字列を送受信するための通信が発生する．本研究では，通常のソケット通信を用いて通信機能を実現した．したがって通信内容が正しく送受信されるかどうかの保証は TCP/IP に負っている．トップサーバ PC が，各 PC の IP アドレスとポート番号を管理する．通信を行うタイミングは以下のとおりである．

(1) ゲート，クレーン，船 PC を立ち上げたとき

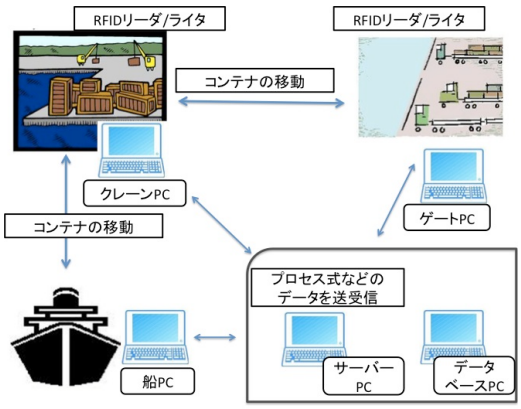

図 1　機器の配置
Fig. 1　Devices location.

(2) プロセス式を分散配置するとき
(3) コンテナがゲートを通ったとき
(4) クレーンがコンテナをつかんだとき

(1) ではトップサーバに IP アドレス，ポート番号を送信する．(2) では各クレーン，ゲート，船 PC にプロセス式自動生成システムからプロセス式を送信する．(3) ではゲート PC からクレーン PC にコンテナ ID を送信する．(4) ではクレーン PC から船 PC に遷移し終わったプロセス式を送信する．

(2) で送信するプロセス式は次に示すとおりである．

- 各コンテナに関するプロセス式
- 各港に関するプロセス式
- 船が通るルートに関するプロセス式

現在は実験で IP アドレスの管理などの便宜のためにトップサーバ PC を用いているが，今後はブロードキャストで接続先の PC の IP アドレスを自動的に確定するよう実装することで，トップサーバを使用しないようにすることを考えている．

5.3　データベースの利用について

本システムでは，コンテナが正しく取り扱われているかどうかの監視，たとえばあるコンテナが正しい船に積み込まれようとしているか，降ろすべき港に積み降ろされようとしているかといった動作の監視は基本的に，そのコンテナに貼り付けた RFID タグに書き込まれた MAC のプロセス式を読み取ることで行う．しかしコンテナの量が多いことから，輸送が始まる前にすべての RFID タグにプロセス式を書き込むことは人手では不可能である．そこでプロセス式生成システムで生成された式を各機器に分散する際，コンテナの個別式に関してのみ，積み込み港のコンテナヤードに設置する PC の持つデータベースに送る．このデータベースは，タグ ID とその ID を持つタグに書き込むべきコンテナ ambient のプロセス式を管理している．

コンテナヤードにコンテナがトレーラに積まれて搬送されてきた際，コンテナヤードのゲートに設置した RFID リーダによってタグの ID を読み取り，その情報をコンテナヤード PC に送信する．これにより，コンテナヤード PC 内のコンテナヤード ambient 内に，その ID に対応したコンテナ ambient が現れる．たとえばその ambient 名を co1 とすると，クレーン PC 内の MAC 式処理系において，`CY[...]` という式が `CY[co1[...]]` に変化する．そしてクレーンによってそのコンテナが船に積まれるとき，クレーン PC に設置した RFID 機器によってそのコンテナに貼り付けられた RFID タグの ID を読み取り，対応するコンテナ ambient 式をそのタグに書き込む．この後のコンテナ取扱いの監視は，すべてここで書き込まれた式を基に行われる．

5.4　クレーン PC とコンテナ搬出入の監視について

コンテナヤードからコンテナが搬出される際，コンテナヤードの出口に設置した RFID 機器によりその搬出動作を検知し，またコンテナ船の入り口部分に設置した RFID 機器によりその搬入動作を検知し，それらの動作の正しさを判定する，というように実装できれば理想的ではあるが，コンテナヤードには出口など存在せず，コンテナ船には入り口など存在しない．そこで本研究では，クレーンに PC と RFID 機器を装備し，クレーンの PC 内に船が持つべき監視システムとコンテナヤードが持つべき監視システムの役割を負わせる．

クレーンがコンテナをつかんだ際，その積み出しがコンテナ ambient の遷移にそったものであるかどうかを判定する．正しければ遷移後の ambient 式をコンテナのタグに書き込む．また，船 ambient 内にコンテナ ambient が入ったプロセス式を生成する．その積み込みがコンテナ ambient の遷移に沿ったものでなければ，その積み込みが誤りであることを示す警告を PC モニタ上に表示する．

すべての搬入が終了したとき，すべてのコンテナ ambient を持った船 ambient 式 `SHIP[co1[...]|co2[...]...]` がクレーン PC の管理する処理系内にできているので，その船 ambient 式を船 PC に送信する．

5.5　RFID

本研究では，コンテナの移動を検知するために，**図 2** に示す RFID 機器を用いる．使用するリーダ/ライタは三菱電機社製の RF-RW004 である．アンテナは同社製の RF-ATCP002 である．RF タグとの交信に，952～954 MHz の周波数を用いており，最大 7 m の読み取りが可能である．本システムでは長距離の通信が必要となるため，UHF 帯を使用するこの RFID リーダ/ライタを選択した．また，使用するタグは同社製の RF–TGM005 である．これは 64 Kbit の書き込み可能メモリ領域を持ち，コンテナ名やコンテナの移動を表す MAC のプロセス式を書き込むのに十分な容量である．コンテナが移動する際には随時，移動先，移動元が持つリーダ/ライタによって，これらの情報が読み取

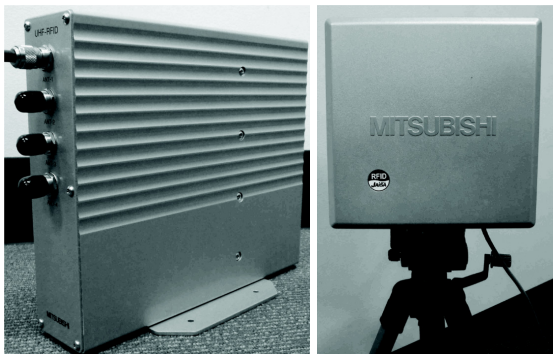

図 2　UHF 帯 RFID リーダ/ライタ
Fig. 2　UHF RFID reader/writer and antenna.

られる．

本システムではタグへの書き込み検査の機能も実装しており，書き込み後すぐに読み取って，内容の確認を行う．さらに，誤って他のタグへの書き込みなどが行われた際の検出機能も実装している．また，タグに書き込まれたプロセス式を基に監視を行うため，タグの読み取りができなければプロセス式の遷移もできず，警告がでることになる．

5.6　プロセス式自動生成システム

本システムは，物流監視に使用するプロセス式を自動的に生成するサブシステムを持つ．これは，貿易書類を読み取り，書類から抽出した情報をもとに MAC で記述されたプロセス式を生成し，物流計画をモデル化するものである．

使用する貿易書類は，送り状，航路表，B/L Instructions，Container Packing List である．これらは Excel 形式で記述されている．これらの Excel ファイルから，POI を用いて物流計画をモデル化するために必要な情報を取得する．POI とは Jakarta プロジェクト [3] の 1 つで，Java で Excel を操作するライブラリである．貿易書類から取得した情報を基に生成されるプロセス式は以下のとおりである．

- コンテナに関するプロセス式：送り状，B/L Instructions，Container Packing List の情報を基に生成される．
- 船に関するプロセス式：航路表の情報を基に生成される．

以上の式を個別式とし，これら個別式の組からなる全体式で物流計画全体をモデル化する．

5.7　プロセス式を分散するシステム

プロセス式自動生成システムにより生成された式は，式分散システムによって各機器へと分散配置される．このシステムは．式を読み込んで解析し，どの港の式なのか，コンテナの式なのか，船の式なのか，といった具合に式を切り分け，それぞれの PC に式を渡す．

式 (1) を切り取った場合，東京港のクレーン PC の船の処理系は式 (5) を持つ．

```
(TK[]
|SHIP[in TK.(load[out SHIP.in CY.in co1]
|open lcomp.out TK.in KB.uload[in co1])],
TK[]|SHIP[in TK.out TK.in KB])
```
(5)

コンテナの式はコンテナヤード PC に送られ，データベースに登録される．また，神戸港のクレーン PC にコンテナヤードの式は送らず船 PC が式を運び，到着した際に船 PC から神戸港のクレーン PC に送信される．

5.8　物流監視システムの動作

本節では，物流監視システムの動作をコンテナ 1 個の輸送を表現する式 (1) を用いて説明する．この式は，5.6 節で述べたように各種貿易書類から自動的に生成され，5.7 節で述べたように各機器へと分散配置される．

まず船が東京港へ入港する．船 PC は，その船の航路などの情報を表す式 (5) を持つ．船が港へと入港した際，入港したという情報をクレーン PC へと渡し，クレーン PC の処理系内で式 (5) を式 (6) へ遷移させる．

```
(TK[SHIP[load[out SHIP.in CY.in co1]
|open lcomp.out TK.in KB.uload[in co1]]], (6)
TK[SHIP[out TK.in KB]])
```

一方トレーラに積まれたコンテナはゲートを通り，コンテナヤードへと運ばれる．この際，ゲート PC ではリーダ/ライタによりコンテナに貼られたタグ ID を読み取り，この ID がクレーン PC へと伝えられる．クレーン PC は，そのタグ ID を基にしてデータベースからコンテナ名と式を読み込み，コンテナヤード ambient の中にコンテナ ambient を入れ式 (7) とする．

```
CY[co1[open load.out CY.in SHIP
   .lcomp[out co1]                    (7)
   |open uload.out SHIP.in CY]]
```

クレーンが船へとコンテナを積み込む際，クレーン PC とつながったリーダ/ライタで，コンテナに貼られているタグの ID を読み取る．そして式 (6) と式 (7) を組み合わせる．このときコンテナが船に入るという遷移ができるのであれば，コンテナの積み込みを承認し，遷移ができないのであれば，それは間違ったコンテナの積み込みであると警告を出す．式 (6) と式 (7) の場合は遷移が可能なので積み込みを承認し，ここからさらに遷移を進め式 (8) となる．

```
(TK[SHIP[out TK.in KB.uload[in co1]
     |co1[open uload.out SHIP.in CY]]],  (8)
TK[SHIP[out TK.in KB]])
```

式 (8) は，コンテナが船へと積まれた状態を表している．そして式 (8) の中のコンテナを表す部分式である式 (9) がタグへと書き込まれる．

```
co1[open uload.out SHIP.in CY]          (9)
```

この例における輸送では，コンテナは 1 つだけなので，船は出港してもよい，という状態になる．船が出航するとき，クレーン PC の船の処理系が持っている式を船 PC へと渡す．このとき，東京港から出るという遷移を行うので，船 PC は式 (10) を持つことになる．

```
(SHIP[in KB.uload[in co1]|co1[]],
SHIP[in KB])
```
(10)

コンテナに取り付けたタグには，行き先などの情報を表す式 (9) が書き込まれており，これによってその後はデータベースに情報を問い合わせる必要がなくなり，タグに書き込まれた内容だけで物流監視を続けることができる．

神戸港に着いた後，船は入港して式 (10) を神戸港のクレーン PC へと送信する．コンテナが積み降ろされる際にタグから式 (9) が読み取られ，クレーン PC の処理系が持つ式に追加して式 (11) とする．

```
(KB[SHIP[uload[in co1]
|co1[open uload.out SHIP.in CY]]],
KB[SHIP[]])
```
(11)

コンテナを船に積むときのように，船から降りるという遷移ができるのであれば，そのコンテナは降ろしてもよいということであり，できないのであれば警告を出す．この例の場合は遷移が可能なので積み降ろしを承認し，コンテナを港へと積み降ろしたところまで遷移を進めた co1[] という式をコンテナのタグに書き込む．またクレーン PC の処理系が持つ式は式 (12) となって監視は終了する．

```
(KB[SHIP[]|CY[co1[]]],KB[SHIP[]])
```
(12)

この例の場合はここで監視は終了するが，このコンテナが神戸港で乗り換えて，他の船でさらに輸送されていく場合，5.9 節で説明するようにコンテナのタグには目的地までの経路を表す式が書き戻されており，コンテナが再び船に積まれる際にタグから式を読み取り，それを基にして監視を続けていく．

5.9　コンテナの乗り換えのモデル化

本研究で対象としている海上物流のハブ・アンド・スポーク・システムでは，コンテナは中継港に到着するたびに，送り状に記載された最終仕向港と輸送経路決定表に基づいて新たな B/L Instructions が作成され，乗り換えのための船が決定される．

本研究では，このような動的なコンテナ輸送経路の決定方法も MAC でモデル化する．例として，名古屋で乗り換えて大阪まで運ばれるコンテナ輸送を表すプロセス式を用いて説明する．

中継港 (名古屋港) のクレーン PC の持つ処理系に，router ambient と呼ぶ式 (13) のような部分式を持たせる．

```
router[
open query.<load_v1_NAGOYA,
SHIP_ShipT_v1,load_lock_NAGOYA,
uload_v1_OSAKA,uload_lock_OSAKA>]
```
(13)

また，式 (14) の中に示すように，コンテナ ambient の中に query ambient，ans ambient と呼ぶ部分プロセス式を持たせる．

```
co1[
  query[out co1.in router
    .(a,b,c,d,e)(
      .ans[out router.in co1.in b
           .out b.out CY.in c.d[out a]
           |in e.out e.out c.in CY
              .f[out a.out CY.in c])]]
|!open ans|!open nextPort]
```
(14)

コンテナが中継港で降ろされタグが読み取られると，クレーン PC において，式 (13) と (14) が処理系内で並列な位置に現れる．query ambient は co1 ambient の外に出て router ambient の中へと入り，query ambient が open され，入出力が起こり，ans ambient 内の各 name a，b，c，d，e に必要な情報が渡される．ans ambient はその情報を持って co1 ambient へと戻り open される．この遷移により，co1 ambient は次に乗る船の名前などの情報を得ることができる．この co1 ambient が元のタグに書き戻され，積み込みの際にはその情報を基に監視が行われる．

5.10　システムの規模

本システムは，Java 言語を用いて構築した．システムの規模は総クラス数 56，総行数 14,000 行程度である．詳細は以下のとおり．

- 監視を行うための処理系：クラス数 33，行数 11,000 程度
- プロセス式自動生成システム：クラス数 18, 行数 1,700 程度
- 分散システム：クラス数 2，行数 700 程度
- RFID の制御：クラス数 2，行数 400 程度
- データベースの管理：クラス数 1，行数 300 程度

6.　実験

本章では，実際の現場での運用を想定して行ったいくつかの実験について述べる．実験は，簡単なコンテナの輸送および乗り換えを想定したものである．ハブ・アンド・スポーク・システムではコンテナの乗り換えを行う．そのため，ただ運ぶだけでなく，乗り換えを想定した実験も必要であると判断し，輸送の基本的なモデルとして下記 4 つの実験を行った．

6.1　実験 1（屋外実験）

1 つ目の実験は東京から神戸港へコンテナを 1 個輸送する最も基本的な輸送を想定したものである．これは**図 3** に

図 3　車に貼り付けたタグを読み取る様子
Fig. 3　Experiment for reading and writing the RFID tag.

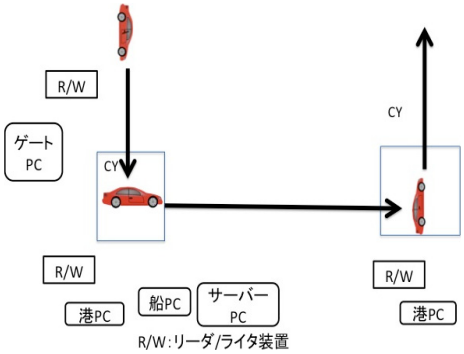

図 4　機器やタグの配置
Fig. 4　Devices and their location.

示すようにコンテナに見立てた車にタグを貼り，リーダ/ライタで読み取りや書き込みを行うことで，屋外における機器の可用性も含め，実際の現場での本システムの実現性を確認する最も基本的な実験である．リーダ/ライタなどの機器の配置は図 4 に示すとおりである．まず自動生成システムにより生成した式 (1) を分散システムにより分割し各 PC へ分散配置する．その後，コンテナに見立てた車を走らせ，1 つ目のリーダ/ライタの前を通過する．これは東京港のコンテナヤードへ入るゲートの通過を想定した動作である．ここでは ID を読み取るだけであり，時速 40 km，距離 2.5 m で読み取ることができた（10 回試行して 10 回とも問題はなかった）．実際の物流の現場において，コンテナを運ぶトレーラがゲートを通過する際の速度は時速 20 km 以下であり，ゲートとの距離も 2.5 m 以下であることから，ハードウェア的には十分実用に耐えられるものであることが分かった．読み取られた ID はクレーン PC へと送信された．

この後コンテナ（車）はコンテナヤードへと移動した．クレーン PC では，読み取られたコンテナの ID を受信し，この ID をキーとしてデータベースとの照合を行い，`CY` ambient の中にコンテナ ambient が入り，式 (7) ができた．

次に 2 つ目のリーダ/ライタの前で車を止めた．このリーダ/ライタはコンテナヤードから船へとコンテナが積まれる間にタグの読み込みと書き込みを行う．このときコンテナの ID を読み取り，その ID に対応する式 (7) と，すでにクレーン PC の中に現れていた船の式 (6) を組み合わせて `in SHIP` という遷移が行われて積み込みが承認され，遷移後のコンテナ式 (9) がタグに書き込まれた．このとき，クレーン PC につながったリーダ/ライタが静止状態，距離 2.5 m で約 5 秒で式を書き込めることを確認した．また，船の式 (10) が船 PC へ送信された．

その後車と船 PC を目的地まで移動させ，船 PC から入港を表す遷移後の式が目的地のクレーン PC に送信された．また，3 つ目のリーダ/ライタでコンテナのタグから式を読み取り，船とコンテナ式を組み合わせた遷移が起こり，積み降ろが承認され，遷移後の式がタグに書き込まれた．

以上でコンテナが東京港から神戸港に到着するまでの輸送の監視が完了した．

6.2　実験 2

2 つ目の実験は，東京港からコンテナを 1 つ積んで出航した後に，名古屋港，大阪港でさらに 1 つずつコンテナを積み，合計 3 つのコンテナを門司港へと運ぶことを想定した屋内実験である．これは MAC を用いることでコンテナの追加が容易となったことを確認するための実験である．屋内実験では，船やクレーンに搭載する予定の PC とタグを人手で移動させることで作業を行った．あらかじめ 3 つのタグを登録しておき，コンテナはすでにコンテナヤードに運ばれているものとした．それぞれのコンテナの個別式は，どの港から積まれ，最終的にどの港で降ろされればよいのか，ということだけ書かれており，途中どの港に寄ったとしても問題ないようになっている．

船 PC はまず東京港へと入港し，コンテナを 1 つ積んだ．その際，コンテナ ambient の遷移と比較し，その積み込みが正しいものであるか確認した後，遷移後の式をタグに書き込んだ．また，コンテナ ambient 式を含む船 ambient 式を船 PC に送った．次に船 PC は名古屋港へと入港し，コンテナが追加され，船へと積まれた．このとき，船 PC が東京から持ってきた全体式に対し，新たに積むコンテナに関する個別式の追加だけを行った．それによって，コンテナ 2 個の監視を問題なく行うことができていることを確認した．さらに，大阪港でも同様のやりとりが行われた後，門司港へと到着した．門司港でコンテナを 3 つ降ろす際，それぞれの ambient 式の遷移を確認し，コンテナをコンテナヤードに搬入して輸送が完了した．

以上で，船が複数の港に寄って，途中で追加された複数のコンテナを運ぶ場合でも，タグへの式の書き込み，読み取りおよびコンテナの輸送の監視が可能であることを確認した．

6.3　実験 3

3 つ目の実験は，船 A が東京港からコンテナを 1 つ積んで出航した後に，名古屋港でそのコンテナを降ろし，そして船 B が名古屋港からそのコンテナを大阪港へと運ぶという船の乗り換えを含む輸送を想定した屋内実験である．これは本システムがハブ・アンド・スポーク・システムでの乗り換えに対応できることを確認するための実験である．この実験ではコンテナの乗り換えの際に式 (13) と (14) を用いた．

東京港での積み込み，出航までの手順は実験 1 と同様である．ただしコンテナのタグに書き込まれた式は query と ans ambient を含む式 (14) である．

船 A の PC とタグを名古屋港のクレーン PC の前に移動させ，船 A が名古屋港に入港した．コンテナがコンテナヤードへと降ろされる際，クレーン PC 内で式 (13) と式 (14) が組み合わさり遷移が生じ，式 (14) 内の query ambient が router ambient 内に入り，ans ambient が co1 ambient 内に戻り，co1 内に次に乗る船の情報が現れたことを確認した．さらにこの co1 ambient がタグに書き込まれたことを確認した．その後，船 B の PC が名古屋港へと入港し，タグに書き込まれたプロセス式を基に遷移が行われて船 B への積み込みが承認された後，コンテナは船 B へと積まれた．そして船 B の PC は大阪港へ移動し，タグの式を確認してコンテナを降ろした．

このように複数の船を使い，コンテナを中継港で積み替えて運ぶという輸送であっても，本システムを用いてその動作の監視が可能であるということが示された．

6.4　実験 4

4 つ目の実験は，誤った動作を本システムが検出できることを示す屋内実験である．東京港からコンテナを 5 つ積んで神戸港へ向かうという輸送を表すプロセス式を用いて監視を行った．コンテナを 3 つ積み込んだだけで船 PC を出航させようとしたとき，**図 5** に示すような，コンテナの積み忘れを示す警告が表示され，誤った動作を検出できることが確認できた．

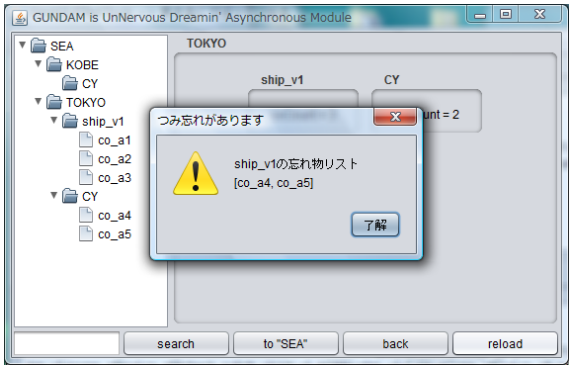

図 5　警告画面
Fig. 5　Warning of a mistake.

7.　結論

本稿では，コンテナを輸送する海上物流を，形式的言語である多重 Ambient Calculus（MAC）[12] を用いてモデル化し，実際のコンテナの取扱いが正しく行われているかどうかをそのモデルに沿って自動的に確認することのできる物流監視システムを提案した．MAC のプロセス式を用いることで，従来の Ambient Calculus のプロセス式を使った物流監視 [11] に比べ，コンテナの追加などで生じる動的な変化に柔軟に対応できることを示した．また，UHF 帯 RFID 機器を使用することで，数メートル離れた場所からコンテナの情報を読み取ることができることを屋外実験により実証した．これにより，本稿で提案するシステムが，今後より実際の物流現場に近い状況で実験可能であることを示すことができた．

ところでこのモデル，すなわち MAC のプロセス式は，物流システムを規定する各種書類から自動生成されるのだが，このモデルに誤りがある場合，自動監視がかえってコンテナの正常な取扱いを妨げてしまうことになる．この問題に対し，我々はすでに MAC を用いたモデル検査システムを構築している [13]．このモデル検査システムは，Ambient Logic と呼ばれる様相論理の論理式により記述された物流システムの持つべき性質，たとえばあるコンテナが必ず目的地に到着するなどといった性質をそのプロセス式が満たすかどうかを検査するものである．この検査を行うことにより，自動生成された MAC のプロセス式が正しいものであるかどうか，あるいはそもそも，物流書類に人為的な誤りが混入していないかどうかを確認することができる．このようにしてその正しさが確認された MAC のプロセス式を本稿で提案した監視システムで使用することにより，より確実な監視作業を行うことが期待できる．

本稿では本システムを使用したいくつかの物流監視実験について述べたが，実際の物流では，より大規模で，より複雑な経路をたどったコンテナ輸送が行われている．本システムの実現性を示すため．今後，そのような物流を規定する書類からプロセス式を生成し，様々な場合を想定した実証実験が必要となる．

本稿で提案したシステムでは海上物流を想定しているが，海上物流の場合は取り扱うコンテナの数が多く，乗り換えなど経路も複雑であり，使用する書類も多岐にわたり，さらに天候などの影響による運航計画の変化など動作が複雑になるからである．このような複雑な動作を MAC によりモデル化できることを示した．したがって，たとえば天候などの影響を受けにくく，物流計画が途中で変化しない他の物流形態に本システムを適用することは容易であると考えられる．

謝辞　本研究は科研費（23500104）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] Clarke, E.M., Grumberg, O. and Peled, D.A.: *Model Checking*, The MIT Press (1999).
[3] Foundation, A.S.: The Apache Jakarta Project, available from ⟨http://jakarta.apache.org/⟩.
[4] 岡田翔太，馬谷誠二，林　奉公，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] 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，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).
[6] 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).
[7] 国土交通省：「メコン地域陸路実用化実証走行試験」—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⟩.
[8] 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⟩.
[9] 今井昭夫：国際海上コンテナ輸送概論，東海大学出版会 (2009).
[10] 山口範高：貿易書類の見方・書き方，同文館出版 (2007).
[11] 森本大輔，樋口昌宏，加藤　暢：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.SIG 10 (PRO33), pp.151–164 (2007).
[12] 樋口昌宏，加藤　暢：物流システム記述のための多重 Ambient Calculus，情報処理学会論文誌：プログラミング，Vol.5, No.2, pp.79–87 (2012).
[13] 樋口昌宏，森田哲平，加藤　暢：多重 Ambient Calculus による物流記述に対する弱双模倣等価性を用いたモデル検査，情報処理学会論文誌：プログラミング，Vol.5, No.3, pp.50–60 (2012).
[14] 布施克彦：貿易書類の基本と仕組みがよ～くわかる本，秀和システム (2005).

橋本 隆弘　（正会員）

2011 年近畿大学理工学部情報学科卒業．2013 年近畿大学大学院総合理工学研究科エレクトロニクス系工学専攻博士前期課程修了．現在，（株）シー・エス・イー勤務．在学中，プロセス代数の研究に従事．

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．

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 年大阪大学大学院博士前期課程修了．（株）富士通研究所勤務，大阪大学基礎工学部助手・講師等を経て，現在，近畿大学理工学部情報学科教授．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．