Vol.2009-MPS-75 No.29
2009/9/11

物流システムに対する Ambient Logic モデル検査システム

加藤 暢[†1] 樋口 昌宏[†1] 植田 直人[†2]

本稿では物流システムの満たすべき性質の形式的な記述体系とそのモデル検査について述べる．物流の世界では，貨物の流通量の増加に伴いコンテナ管理の重要性が高まっている．我々は Ambient Calculus による物流システムの記述と，実際の物流がその記述どおりに行われているかを監視するシステムを開発した．この監視システムによる物流管理の正しさを確認するためには，書類から生成されたプロセス式そのものの正当性を確認する必要がある．しかし，プロセス式生成システムで生成されるプロセス式は一般に複雑なものになり，さらに非決定的な動作も多くこの確認作業を人手で行うのは困難であると考えられる．本稿では，この課題を解決するために，物流システムに要求される，“ いつか必ず貨物は目的地に輸送される ”というような性質を Ambient Logic の論理式で表現し，その性質をプロセス式がすべて満たしていることを形式的に検査する手法を提案する．遷移グラフの各経路が論理式を満たしていることを調べることで検査を行う．提案手法に基づくモデル検査システムを構築し検証実験を行った結果についても述べる．

An Ambient Logic Model Checking System For Freight Systems

ToruKato,[†1,?] Masahiro Higuchi [†1]
and Naoto Ueda[†2]

We present a variant of Ambient Logic to describe desired properties of freight systems and a model checking algorithm for the logic. In the field of distribution, the increase in cargo handling errors is a serious problem as the amount of freight circulation increases. We study a freight inspection system based on Ambient Calculus and we have already built a prototype system. The system can derive process formulas from several freight documents. The System also checks that the actual freight movement conforms the derived formula. To show the correctness of freights transportation, we have to ensure the validity of process formula. We provide an Ambient Logic to present desired properties of freight systems, such as “ every cargo will be eventually transported to its destination ”. We also present an algorithm to show process formula satisfy Ambient Logic formula by exploring state transition graph. We conducted a verification experiment using the model checking system.

1. はじめに

近年，物流の世界では，貨物の流通量の増加に伴う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についてさまざまな方法が模索されている[8),9)]．我々は物流で用いられているさまざまな貿易書類に基づいて，物の動きを自動的に監視できるシステムに関する研究を行っている．貿易書類そのままでは形式的な取り扱いが困難である．そこで我々は，それらの書類に基づき Ambient Calculus[2)] のプロセス式を生成し，実際の物流がそのプロセス式の記述通りに行われているかを監視する方法を考案した．すでに送り状，B/L Instructions，Container Packing List[7)] といった実際に貿易で使われる書類を元に自動的にプロセス式の生成を行い，RFID で検知した物の移動と，プロセス式の遷移とを関連付けることで記述通りの貨物輸送が行われていることを監視するシステムを構築している[11)]．ただしこのプロセス式が，物流システムを正しくモデル化できているということは示されていない．監視システムによる物流管理の正しさを示すためには，書類から生成されたプロセス式そのものの正当性を確認することが必要である．

しかし，プロセス式生成システムで生成されるプロセス式は一般に複雑なものになり，さらに非決定的な動作も多く，この確認作業を人手で行うのは困難であると考えられる．この課題を解決するために，本稿では物流システムに要求される，“いつか必ず貨物は目的地に輸送される” というような性質を様相論理の一種である Ambient Logic[3)] を用いて記述し，生成されたプロセス式が記述した性質すべてを満たしていることを形式的な手法で検査する手法を提案する．本稿では物流システムの満たすべき性質の記述体系とそのモデル検査について述べる．また，これを用いたモデル検査システムと検証実験の結果についても述べる．

2. 物流システムのモデル化のための Ambient Calculus

Ambient Calculus は，MicrosoftResearch 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ための言語である．

†1 近畿大学理工学部情報学科
Kinki University School of Science and Engineering Department of Informatics
†2 株式会社 JSOL
JSOL Corporation

2.1 構文規則

本節では，物流システムのモデル化のために必要となる AmbientCalculus の subcalculus を同定する．モデル化の対象となるものは，海，港，船，コンテナヤード，コンテナの 5 種類のみとする．これらの物を，一意の名前を持った ambient で表現する．また，これらの物の性質に関しては，移動のみをモデル化するため，物と物との間の通信動作を表現する入出力は不必要である．物の移動に関しては，船やコンテナの目的地までの有限回の移動をモデル化するため，再帰は不必要である．現実の物流システムにおいて，輸送の途中で物が増えることはあり得ないので，replication は不必要である．ambient を消滅させる *open* は，物の移動を制御するために必要となる．以上の分析より，本研究で使用する Ambient Calculus の構文規則を定義 2.1 で与える．

定義 2.1 (構文規則)

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

遷移規則に関しては，文献 2) にある定義をそのまま使用する．遷移規則に基づく式の遷移を，コンテナ船 ($SHIP$) の中に存在しているコンテナ (CO) が，船の中から出ていき，さらにコンテナヤード (CY) の中に入ることを記述した式を例に説明する．

$$SHIP[\ CO[\ out\ SHIP.in\ CY\ ]\ ] \mid CY[\ ] \tag{1}$$
$$\xrightarrow{out\ SHIP} SHIP[\ ] \mid CY[\ ] \mid CO[\ in\ CY\ ] \tag{2}$$
$$\xrightarrow{in\ CY} SHIP[\ ] \mid CY[\ CO[\ ]\ ] \tag{3}$$

$$PORT[\ SHIP[\ open\ ctrl.out\ PORT \mid ctrl[\ ]]\ ] \tag{4}$$
$$\xrightarrow{open\ ctrl} PORT[\ SHIP[\ out\ PORT\ ]\ ] \tag{5}$$

式 (1) では $SHIP$ の中に CO があり，最初の遷移が行われた式 (2) では $SHIP$ の中にあった CO が $SHIP$ の外へと移動し，次の遷移が行われた式 (3) では，CO が CY の中へと移動した状態になっている．他の動作として，ambient の境界を消滅させる *open* がある．式 (4) において，$SHIP$ の中の *open ctrl* は，$SHIP$ が港 ($PORT$) の外に移動するのを妨害しており，この *open ctrl* に並行して $ctrl[\ ]$ が存在するので，遷移が行われた後の式 (5) では *open ctrl* が消滅し，$SHIP$ が出航できる状態になっている．上記の式の中で，$SHIP$ はコンテナ船そのものを，CO はコンテナそのものを，CY はコンテナヤードそのものをそれぞれ表す ambient である．そして，それぞれの親子関係は，式 (1) ではコンテナ船とコンテナヤードが並列に存在し，コンテナ船の中にコンテナが入っている事を表している．

3. Ambient Calculus を用いた物流監視システム

ここでは我々が開発している，貨物の運送が輸送計画に沿って行われているかどうかを監視する Ambient Calculus を用いた物流監視システムについて述べる[11)]．

3.1 プロセス式生成システム

物流システムを表現するプロセス式を自動生成するために，どの船でどこの港からどこの港まで貨物を輸送するかを記述している送り状，コンテナの情報を記述している B/L Instructions と Container Packing List という 3 種類の貿易書類と，船の航路表を用いる．プロセス式は物を表現する ambient ごとに分かれて生成される．まず送り状から荷物を積み込む港と積み下ろす港の情報とコンテナ船の名前を取得する．B/L Instructions と Container Packing List からコンテナ情報を取得する．物を表現する ambient に関する必要な情報を読み込んだ後,「すべてのコンテナを積み込んでから船が出港する」等の制約を表現するための制御用 ambient を追加する．以上より，図 1 のような物流を記述した，Ambient Calculus 式が生成される．図 1 は式全体を表示しているが，実際に物流監視を行うときは，物を表現する ambient ごとにプロセス式を分散処理させることで監視を行う．またプロセス式生成システムでは，個々の物流を記述対象としているため，生成されたプロセス式には再帰は含まれない．

ところで，図 1 中の「$checkin(SHIP\_v1)\ CY$」は，
$in\ SHIP\_v1.out\ SHIP\_v1.in\ CY$ の略記,「$checkout(load\_v1)\ CY$」は，
$load\_v1\_TOKYO\ .outload\_v1\_TOKYO.out\ CY$ の略記である．前者は港 $TOKYO$ に船 $SHIP\_v1$ がついたことをコンテナに知らせるために CY に入る制御用 ambient $load\_v1\_TOKYO$ の動作を表している．後者は，その制御用 ambient がコンテナヤード

$$
\begin{array}{l}
SEA[\ KOBE[uload\_v1\_KOBE[in\ SHIP\_v1] \mid CY[]] \\
\quad \mid SHIP\_v1[in\ TOKYO.\underline{open\ lcomp\_a1}.out\ TOKYO.in\ KOBE] \\
\quad \mid TOKYO[\ load\_v1\_TOKYO[checkin(SHIP\_v1)\ CY] \\
\qquad \mid CY[\underline{CO\_a1}[\ \underline{checkout(load\_v1)CY} \\
\qquad\qquad \underline{.in\ SHIP\_v1.lcomp\_a1[out\ CO\_a1]} \\
\qquad\qquad \underline{\mid checkout(uload\_v1\_KOBE)SHIP\_v1.in\ CY]} \\
\qquad] \\
\quad] \\
]
\end{array}
$$

図 1 物流システムを記述したプロセス式

CY に入ってきたことを知った後にコンテナ CO_a1 が CY から出るという動作を表している． 図 1 のプロセス式はコンテナを 1 個 $TOKYO$ のコンテナヤードから $KOBE$ のコンテナヤードへ運ぶ物流を表しており，コンテナを増やす場合は，下線を引いた部分がその数に応じて増加する．

なお，Ambient Calculus では，構文上 name に現れる大文字と小文字の区別に意味はない．しかし本稿で提案するプロセス式生成システムは，物を表現する ambient には必ず大文字で始まる name をつけ，制御 ambient では小文字で始まる name をつけるという制約を導入する．これは，次の 3.2 節で述べるように，物流システムを監視する際にプロセス式を遷移させるタイミングが，物を表現する ambient と制御用 ambient では異なるからである．

3.2 物流と式の遷移との関連付け

監視システム内では，capability を実行するタイミングが物を表す ambient と制御用 ambient では異なる．監視対象の動作を表す物を表す ambient では，実際の物が動いたことを確認した後に，その動作に対応する capability が実行可能な場合のみ遷移が行われる．そのような capability が存在しない場合は，物の移動が本来の物流計画に沿っていないことになり警告を発する．一方，制御 ambient は遷移可能になれば直ちに遷移させる．そのため物を表す ambient と制御用 ambient に構文上の区別を与えた．

物を表現する ambient の遷移と現実のものの移動を関連付けるには，RFID を用いる．まず RF タグをコンテナやコンテナ船に取り付ける．各タグごとにその物を表現する ambient のプロセス式が書き込まれている．そして物が出入りする所に設置されたリーダ/ライタが，物の移動時に RF タグを感知し移動が可能かを監視する．可能であれば RF タグのプロセス式を書き換え，不可能であれば警告を行う．また実際に関連付けを行うには，プロセス式を解釈し，遷移させる処理系が必要になるが，この処理系は HORB を用いてすでに実装されている[6]．このように，実際の物とプロセス式の関連付けを行うことで，物の流れを監視することができる．

4. 物流システムのための Ambient Logic

4.1 検証対象となる性質

物流システムのための限定的な Ambient Logic を導入するために，物流システムの持つ性質を考え，その性質を Ambient Logic で記述するのに必要になる演算子を考える．プロセス式の満たすべき性質として物流システムの所期の性質を満たしていることと，現実の世界では起こりえないことを記述していないことの 2 つを挙げることができる．物流システムの所期の性質はプロセス式を生成するために使用する貿易書類から読み取ることができる．本研究で対象とする上記のような性質を，以下にそれぞれ p1〜p3，s1〜s4 で示す．

物流システムの所期の性質

- p1 いつか必ず貨物（コンテナ）は目的地に輸送される．
- p2 特定の場所以外での貨物（コンテナ）の積み下ろしは行われない．（不正な貨物の移動の禁止）
- p3 コンテナ船には決められた貨物（コンテナ）が積み込まれる．

物理的には起こり得ない事象の排除

- s1 物は移動過程で消えることはない．
- s2 海，港，コンテナヤードは，移動することはない．
- s3 コンテナ船は，海と港の間のみを移動する．
- s4 コンテナに別のコンテナが入ることはない．

4.2 物流システムのための Ambient Logic

ここでは上記の性質を表現するのに必要になる様相記号を考える．p1 を表現するためなどに，一般的な時制論理で用いられる “いつかは”を表現する *sometime modality* や “常に”を表現する *everytime modality* が必要である．また Ambient Logic 特有の，プロセス式中のある ambient 内のプロセス式がある性質を満たしていることを表現する *location* やプロセス式の部分式がある性質を満たしていることを表現する *somewhere modality* が

必要である．これらを考慮し， Ambient Logic のサブセットを用いることにした．文献 3) で導入された 17 個ある Ambient Logic 特有の演算子のうち *composition*, *location*, *somewhere modality* の 3 個を使うことで，示したい性質を記述することができ，モデル検査システムの実装が比較的容易になる．本稿で使用する記号の意味についていくつか説明する．*composition* $P \vDash \mathcal{A} \mid \mathcal{B}$ は，P が $\mathcal{A}$ という論理式を満たすプロセスと，$\mathcal{B}$ という論理式を満たすプロセスが並列に存在するプロセス式であることを示している．*location* $P \vDash n[\mathcal{A}]$ は，P は $\mathcal{A}$ という論理式を満たすプロセスをもつ n という ambient が階層構造の最上位に存在するプロセス式であることとを示している．*somewhere modality* $P \vDash \Diamond\mathcal{A}$ は，P が持つ階層構造のどこかに $\mathcal{A}$ という論理式を満たすプロセスが存在するプロセス式であることを示している．

5. モデル検査システム

5.1 検査システム

本検査システムでは，与えられたプロセス式に対して初めに図 2 のようなプロセス式の遷移グラフを作成する．遷移グラフの各ノードはノードを識別するためのノード ID，遷移経路を知るための親ノード ID，そして図 3 に示すようなプロセス式の構造を表す構文木を持っている．構文木の各ノードは，ambient の名前，各 ambient を識別するための ID，親子関係等を識別するための親 ambient の ID，*capability action* のリストを持っている．この構文木の *capability action* のリストと木の親子関係を見ることで，可能な遷移を見つけ，構文木を遷移させることで図 2 のような遷移グラフの作成を行う．この遷移グラフを深さ優先で探査し，各ノードの構文木に対して *location* や *somewhere modality* のような ambient の空間的な検査を木の親子関係を見ながら行い，*sometime modality* や *everytime modality* のような時間的な検査を経路を見ながら行う．

例えばプロセス式生成システムにより生成されたプロセス式 (6) のモデル検査するために遷移グラフを作る場合，まずプロセス式を構文解析し構文木を作る．この構文木に各種ノード情報を付加する．これが遷移グラフにおける初期ノード $S0$ になる．

$$SEA[\ \ SHIP1[\ in\ PORT \mid CO[\ out\ SHIP1\ .\ in\ CY\ ]\ ] \mid PORT[\ CY[\ ]\ ]] \qquad (6)$$

構文木の各ノードが持つ *capability action* のリストから，そのノードで実行可能な遷移を見つけ，遷移後のノードを作成する．図 3 の場合，ambient ID 0 の $SHIP1$ の持つ

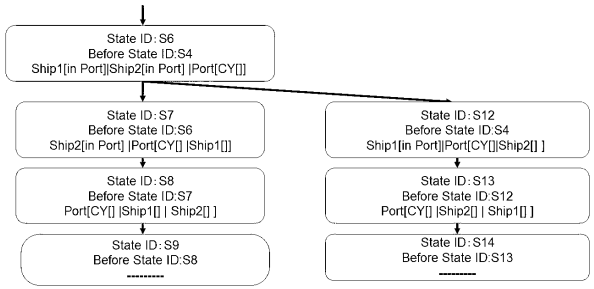

図 2 遷移グラフ

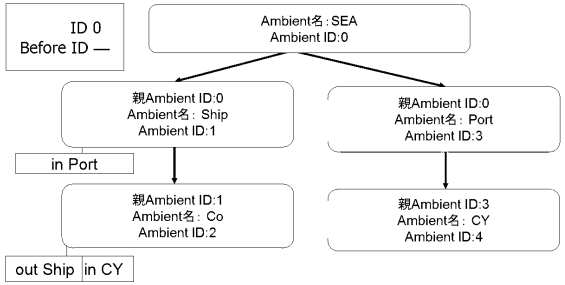

図 3 構文木

capability action は *in PORT* なので，この ambient の兄弟に *PORT* という ambient が存在するか検査する．この例の場合は *PORT* があるので *capability action* を削除し，*SHIP* の親 ID を *PORT* の ID である 3 に変えることで，図 4 のように遷移後の構文木に変える．*location* や *somewhere modality* などの空間的な検査は，指定された場所に ambient の階層構造があるかを構文木をたどることで検査することができる．

5.2 検証内容

輸出港 *PORT_A* から港 *PORT_C* を経由し輸入港 *PORT_B* にコンテナ *CO* を輸送する物流システムを例に，本検証系の対象とする性質を説明する．プロセス式の満たすべき性質を以下のような式で表現する．

- いつか必ず *CO* は *PORT_B* に輸送される．

$$P \vDash \Box\Diamond\Diamond PORT\_B[CY[CO[T] \mid T] \mid T]] \qquad (7)$$

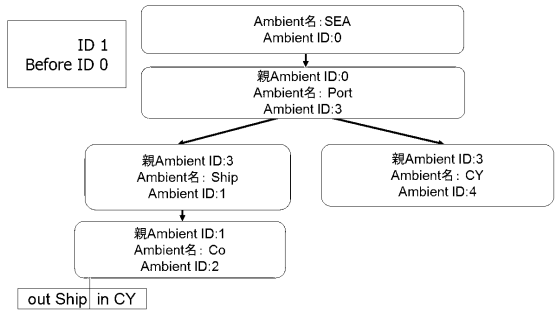

図 4 遷移後の構文木

(7) 式の $CY[CO[T] \mid T]$ の部分はコンテナヤード CY の中にコンテナ CO が存在しており，また $CO[T] \mid T$ でコンテナヤード CY の中に，さらにコンテナ CO 以外が存在してもよいことを示している．$\Diamond PORT\_B[CY[CO[T] \mid T] \mid T]$ は，プロセス式のどこかで $PORT\_B[CY[CO[T] \mid T] \mid T]$ という階層構造が存在することを示しており，この式に $\Box\Diamond$ をつけることで，どのような遷移が行われたとしてもどこかでその状態が必ず成り立つことを示している．p1 の性質は，コンテナが輸入港のコンテナヤードにあればいいので (7) 式で表すことができる．

- $PORT\_A$,$PORT\_B$ 以外では CO の積み下しは行われない．

$$P \vDash \Box(\neg\Diamond PORT\_A[SHIP[T] \mid T] \wedge \neg\Diamond PORT\_B[SHIP[T] \mid T] \Rightarrow \neg\Diamond(SHIP[T] \mid CO[T])) \quad (8)$$

p2 の性質は，コンテナを積み込む直前または積み下した直後は $(SHIP[T] \mid CO[T])$ が成り立ち，積み込み，積み下ろしができるのは，それぞれ輸出港と輸入港だけなので “輸出港，輸入港以外ではコンテナの積み下しは行われない” と言い換えることができ (8) 式で表すことができる．

- $SHIP$（コンテナ船）には指定された CO のみが積み込まれる．

$$P \vDash \Box(\neg\Diamond PORT\_A[\Diamond CO[T]] \wedge \neg\Diamond PORT\_B[\Diamond CO[T]] \Rightarrow \Diamond(SHIP[CO[T] \mid T])) \quad (9)$$

p3 の性質は，輸出港か輸入港のどこかにコンテナがない場合には必ず指定されたコンテナ船にコンテナが積み込まれていることを要求しているので (9) 式で表現できる．この 3 つの式で，物流システムの所期の性質を Ambient Logic で表現できる．

5.3 状態空間爆発問題

遷移グラフには，どのコンテナを先に移動させるかによる枝分かれが生じる．しかもコンテナ数は一般に数百個から数千個のオーダーなので，その枝分かれ数は膨大になり，その枝分かれに沿った検証は実際的には不可能である．実際，6 個のコンテナを輸送するプロセス式の場合でさえ状態空間爆発が発生し，数 GB のメモリのを持つ計算機では検証不可能であった[10]．

5.4 Ambient Calculus のプロセス式の性質を利用した partial order reduction

このような問題を解決するために，partial order reduction と呼ばれる方法がある[1),4)]．文献 4) で述べられている方法は次の通りである．遷移グラフにおいて各ノード s からの遷移を，そのノードで実行可能な遷移の集合 $enabled(s)$ の部分集合 $ample(s)$ に制限することによって，遷移グラフの状態空間爆発を抑制することができる．最小の $ample(s)$ を正確に求めるためには全状態空間の探索が必要となるため，通常は何らかのヒューリスティクスに基づいて，なるべく小さい $ample$ 集合を求める方法がとられる．本研究では Ambient Calculus のプロセス式の持つ性質に着目して，$enabled(s)$ とプロセス式の持つ情報のみから $ample(s)$ を決定できる方法考案し，その方法に基づく検査システムの実装を行った．

5.5 プロセス式の均質性を利用した partial order reduction

コンテナ CO_a1 と CO_a2 の積込港と積降ろし港が同じ場合，それぞれのコンテナを表す ambient 名が異なるだけで，ambient 内の capability は全く同じである．また，それぞれのコンテナに対応した制御用 ambient も同じである．このように本研究で扱っている物流システムは，コンテナの数は多いがその多くにはこのような均質性が見られるという特徴を持つ．したがって，プロセス式の中のこのような均質性さえ確認できれば，これらの ambient の非決定的な動作による違いから生じる複数の遷移は，どれか 1 つを代表として残し，他の遷移は遷移グラフから削除してもモデル検査には影響しない．つまり，プロセス式中の均質性を利用した partial order reduction が可能となる．本研究ではこの方法に基づく状態数の削減手法も検査システムに取り入れた．

6. 実験結果

図 1 で示したような，ある港からある港へコンテナを輸送する式を用意した．但し輸送されるコンテナ数は 3〜150 とした．つまり，図 1 における CO_a1 が，CO_a1〜CO_a150 であるような式である．このプロセス式に対し，前節で示した手法を用いて簡約化した遷移グラフ

を作りながら，要求される性質を表す論理式の論理積を，プロセス式が満たすかどうかという検査実験を行った．実験環境は次の通り．OS:Vine Linux 5.0β2 64bit 版，CPU:Core2Duo 3GHz，メモリ:4GB，JDK1.6.10．結果は表 1 のようになった．

	遷移グラフのノード数			
輸送するコンテナ数	3 個	10 個	100 個	150 個
全状態探査 (秒)	2475 (26.6)	計測不能	計測不能	計測不能
ample(s) のみ (秒)	413 (0.6)	計測不能	計測不能	計測不能
対称性のみ	56 (0.2)	8,170 (11.1)	計測不能	計測不能
ample(s) と 対称性を併用	36 (0.2)	246 (1.1)	17,131 (38 分 53 秒)	41,795 (4 時間 35 分)

表 1 状態数と検査時間

例えば 150 個の欄では，150 個のコンテナ輸送を表す式が所期の性質を満たしていることを，本稿で示した二つの partial order reduction を併用すると約 4 時間 35 分 (16,503,219ms) で検証できたことを示している．計測不能と書かれた個所は，5 時間以上経過しても結果が返ってこなかったことを示している．

物流書類に誤りがあった場合，あるいはプロセス式生成システムに不具合があった場合，論理式を満たさないことが検出され，検査が終了する．物流書類に誤りがあった場合，あるいはプロセス式生成システムに不具合があった場合いずれの場合も式中の誤りの部分の特定は容易であり，その後その部分に対し目視による物流書類の検査を行うことができる．

7. 結　　論

本稿では，Ambient Calculus によりモデル化された物流システムに対するモデル検査システムのための公理系，モデル検査システム，及び実験結果について述べた．本モデル検査システムにより，プロセス式生成システムから物流書類をもとに生成されたプロセス式が所期の性質を満たすことを形式的に示すことが可能となった．

また，単にプロセス式生成システムの生成方法の正しさを示すだけでなく，物流書類に内在する誤りに対しても，輸送が始まる前に訂正を促すことが期待できる．これは，輸送業者の意図，例えば貨物はどこでどの港で積み込み積み下ろすのか等を反映した論理式を与えることができれば，その式を用いて物流書類から生成されたプロセス式を検査することで可能となる．ただし現状では，実際的な時間で検査できるプロセス式が，コンテナ百数十個にとどまっており，更なる検査アルゴリズムや実装方法の効率化が必要である．

本稿では１つの港から１つの港へのコンテナを輸送する場合を想定した単純な実験のみの結果を示したが，数 10 個は港 A で，数 10 個は港 B で積み下ろす，あるいは途中の港で数 10 個をさらに積み込む，積み替える等の複雑な経路を想定した実験を今後予定している．

参 考 文 献

1) Affeldt, R. and Kobayashi, N.: Partial Order Reduction for Verification of Spatial Properties of Pi-Calculus Processes, *Proceedings of the 11th International Workshop on Expressiveness in Concurrency (EXPRESS 2004)* (2004).
2) Cardelli, L. and Gordon, A.D.: Mobile Ambients, *LNCS*, Vol.1378, pp.140–155 (1998).
3) Cardelli, L. and Gordon, A.D.: Any time Anywhere Modal logics for Mobile Ambients, *POPL'00*, Proceedings of the 2000 ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp.365–377 (2000).
4) Clarke, E., Grumberg, O. and Peled, D.: *Model Checking*, MIT Press (2000).
5) 片山立志：よくわかる貿易実務入門，日本能率協会マネジメントセンター (2007).
6) Kato, T., Ikeda, D. and Okada, Y.: The Implementation of Ambient Calculus with HORB for Mobile Agents, *Proc. of The 7th World Multiconference of Systemics, Cybernetics and Informatics*, Vol.II, pp.367–372 (2003).
7) 山口範高：貿易書類の見方・書き方，同文舘出版 (2007).
8) 国土交通省：メコン地域陸路実用化実証走行試験，*http://www.meti.go.jp/press/20071018006/press_mMM.pdf* (2007).
9) 国土交通省：海上貨物追跡タグシステム (MATTS)，*http://www.mlit.go.jp/report/press/port02_hh_000006.html* (2008).
10) 植田直人，　加藤暢，樋口昌宏：Ambient Calculus による物流システム記述に対するモデル検査，FIT2008 第 7 回情報科学技術フォーラム，pp.13–16 (2008).
11) 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，Vol.48, No.SIG 10(PRO33), pp.151–164 (2007).