RA-005

Ambient Calculus による物流システム記述に対するモデル検査
Model Checking for Ambient Calculus Description of Freight System

植田 直人† 加藤 暢‡ 樋口 昌宏†
Naoto Ueda Toru Kato Masahiro Higuchi

1 はじめに

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うりに行われているかを検査するシステムの研究を行っている。すでに送り状、B/L Instructions 、Container Packing List 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 で感知した物の移動と、プロセス式の遷移とを関連付けることで記述どうりの貨物輸送が行われていることを検査する検査システムを構築している [1]。この検査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。この課題を解決するために、生成されたプロセス式が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” のような性質をすべて満たしていることを形式的な手法で検査することを考えた。本論文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。

2 Ambient Calculus

Ambient Calculus[2] 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。この特徴から物流システムの持つ階層構造を簡潔に表現することが出来る。さらに、様相論理の一種である Ambient Logic の公理系を用いて、プロセス式そのものが意図した性質を持っているかどうかを検証することが可能である。

2.1 構文規則

Ambient Calculus の構文規則は表 1 のように定義されている。

表 1 構文規則

$P, Q ::=$ *processes*		$M, N ::=$ *capabilities*	
(νn)	*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
$M.P$	*capability action*	ϵ	*null*
$(x).P$	*input action*	$M.N$	*path*
$\langle M \rangle$	*output action*		

2.2 プロセス式の遷移例

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 (*cont*) 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。

$$ship[\ cont[\ out\ ship.in\ cy\ ]\ ]\ |\ cy[\ ] \quad (1)$$

$$\xrightarrow{out\ ship} ship[\ ]\ |\ cy[\ ]\ |\ cont[\ in\ cy\ ] \quad (2)$$

$$\xrightarrow{in\ cy} ship[\ ]\ |\ cy[\ cont[\ ]\ ] \quad (3)$$

out は ambient を今いる ambient の外へ移動させる能力、*in* は ambient を他の ambient の中へ移動させる能力を表現している。式 (1) では *ship* の中に *cont* があり、最初の遷移が行われた式 (2) では *ship* の中にあった *cont* が *ship* の外へと移動し、次の遷移が行われた式 (3) は *cont* 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* などがある。

上記の式の中で，*ship* はコンテナ船そのものを、*cont* 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。このように、Ambient Calculus を用いることにより記述対象の階層構造を適切に、かつ直観的に表現することが出来る。これ以降、コンテナやコンテナ船などを表す ambient について、“物を表現する ambient” という表現を使う。

3 Ambient Calculus を用いた物流検査システム

我々は Ambient Calculus を用いて、貨物の運送状態が正しいかどうかを検査するシステムの開発を行っている [1]。本節では、この物流検査システムについて述べる。

3.1 物流システムのモデル化

一荷主の貨物で一つのコンテナすべてを使うコンテナ船輸送である FCL 海上貿易を物流システムの例に挙げモデル化の説明を行う。FCL 海上貿易における物の流れは、まずコンテナヤード内にすでに荷物が積み込まれたコンテナが荷主によって運び込まれている状態があり、その港にコンテナ船がやってくると、コンテナはコンテナヤードから運ばれコンテナ船へと船積みされる。その後コンテナ船は港を出港し、さまざまな港を経由し目的地となる港に向い、港に到着したコンテナ船は積んでいたコンテナをコンテナヤードに運び、受取人はそのコンテナを受け取るという流れで行われる。このように物流システムは、小さなパッケージを含むパッケージが、さらに大きなパッケージに収容されるという階層構造を持っていることと、その構造が動的に変化していくという特徴を持っており、Ambient Calculus の特徴と共通する点が多く、比較的容易に物流システムをモデル化することができる。

† 近畿大学大学院総合理工学研究科
‡ 近畿大学理工学部情報学科

3.2 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と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。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。

3.3 物流と式の遷移との関連付け

生成されたプロセス式では物を表現する ambient と制御を表現する ambient を構文上では区別しない。しかし物流システムを検査する際にプロセス式を遷移させる場合には、それぞれ区別する必要がある。それは、制御を表現する ambient は遷移可能なときであればいつ遷移してもかまわないが、物を表現する ambient の遷移は実際に物が動いたタイミングでのみ遷移しなければならない。そのため物を表現する ambient の遷移と現実のものの移動を関連付ける必要がある。そこで生成されたプロセス式と実際の物の移動の関連付けには 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、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検査する。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 [3]。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、物流検査を行うことができる。

4 物流システムのための Ambient Logic

プロセス式生成システムにより生成された式が物流システムの所期の性質を満たしているか検証することを考える。このことを検証するために、様相論理の一種である Ambient Logic[4] を用いることが考えられる。しかし、Ambient Logic に対応した検証系の構築は容易ではないので、物流システムのための限定的な Ambient Logic の導入を考えることにした。

4.1 検証対象となる性質

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。それらの例を以下にそれぞれ 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* やプロセス式のどこかで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、以下に示す Ambient Logic のサブセットを用いることにした。*somewhere modality* を表現する $P \downarrow P'$ は入れ子構造を示しており、$P \equiv n[P']$ のようにプロセス P の一段下にプロセス P' が含まれていることを示している。また $\downarrow^*$ は、$\downarrow$ の反射的推移的閉包を表している。

Logical Formulas

$\eta\ \mu$ a name n or variable x

$A\ B\ C ::=$

T	*true*
$\neg A$	*negation*
$A \vee B$	*disjunction*
$A \mid B$	*composition*
$\eta[A]$	*location*
$\Diamond A$	*sometime modality*
$\Diamond A$	*somewhere modality*

Satisfaction

Π	*sort of processes*
Φ	*sort of formulas*
Λ	*sort of names*

$P \downarrow P'$ iff $\exists n, P''.P \equiv n[P'] \mid P''$

$\downarrow^*$ is the reflexive and transitive closure of $\downarrow$

$\forall P \in \Pi \quad P \vDash T$

$\forall P \in \Pi, A \in \Phi \quad P \vDash \neg A \triangleq \neg P \vDash A$

$\forall P \in \Pi, A, B \in \Phi \quad P \vDash A \vee B \triangleq P \vDash A \vee P \vDash B$

$\forall P \in \Pi, A, B \in \Phi \quad P \vDash A \mid B \triangleq \exists P', P'' \in \Pi.P \equiv P' \mid P'' \wedge P' \vDash A \wedge P'' \vDash B$

$\forall P \in \Pi, n \in \Lambda, A \in \Phi \quad P \vDash n[A] \triangleq \exists P' \in \Pi.P \equiv n[P'] \wedge P' \vDash A$

$\forall P \in \Pi, A \in \Phi \quad P \vDash \Diamond A \triangleq \exists P' \in \Pi.P \rightarrow^* P' \wedge P' \vDash A$

$\forall P \in \Pi, A \in \Phi \quad P \vDash \Diamond A \triangleq \exists P' \in \Pi.P \downarrow^* P' \wedge P' \vDash A$

Derived Connectives

F	$\triangleq \neg T$	*false*

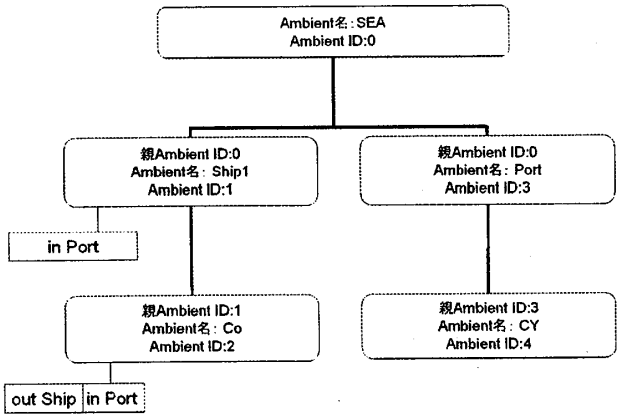

図 1　構文木

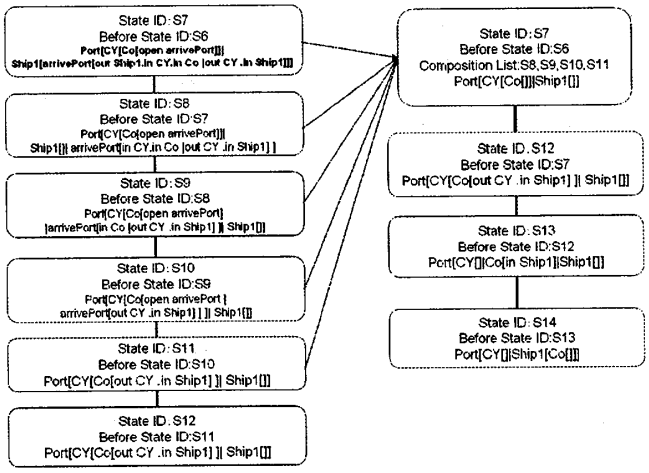

図 2　遷移グラフの縮小

$A \wedge B \quad \triangleq \neg(\neg A \vee \neg B) \quad$ *conjunction*
$A \Rightarrow B \quad \triangleq \neg A \vee B \quad$ *implication*
$\Box A \quad \triangleq \neg\Diamond\neg A \quad$ *everytime modality*

Some Expanded Definitions

$\forall P : \Pi \quad P \vDash T$
$\forall P : \Pi, A, B : \Phi \quad P \vDash A \wedge B$ iff $P \vDash A \wedge P \vDash B$
$\forall P : \Pi, A, B : \Phi \quad P \vDash A \Rightarrow B$
iff $P \vDash A \Rightarrow P \vDash B$
$\forall P : \Pi, A : \Phi \quad P \vDash \Box A$
iff $\forall P' : \Pi . P \rightarrow^* P' \Rightarrow P' \vDash A$

5　モデル検査システム

本節では 4 節で示した公理系に基づく、本研究で構築したモデル検査システムについて述べる。

5.1　検査システム

本検査システムでは、初めにプロセス式の遷移グラフを作成する。グラフの各ノードは図 1 に示すようなプロセス式の構造を表す構文木であるこ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 の空間的な検査を行いながら、*sometime modality* や *everytime modality* のような時間的な検査を行う。

例えばプロセス式 (4) の遷移グラフを作る場合、まず構文木を作る。これが遷移グラフにおける初期ノード $S0$ になる。

$$SEA[\ Ship1[\ in\ Port \mid Co1[\ out\ Ship1\ .\ in\ CY\ ]\ ]\ ] \mid Port[\ CY[\ ]\ ]\ ] \quad (4)$$

構文木の各ノードは *capability action* のリストを持っており、そのノードの次に到達可能な遷移を見つけることができる。*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。実際の物流では多くの船やコンテナを取り扱うことになるが、一隻の船で数個のコンテナを輸送する書類から生成されるプロセス式でも、到達可能なノード数は 10 万を超える。この遷移グラフを用いた検査を効率的に行うために、できるだけノード数を減らし遷移グラフの簡約化を行う必要がある。遷移グラフを簡約化するための方法として構造合同と *stuttering equivalent*[5] に基づく簡約化を行う。まず構造合同に基づく簡約化を考える。

$$Ship1[\ in\ Port] \mid Ship2[\ in\ Port] \mid Port[\ CY[\ ]\ ] \quad (5)$$

(5) 式は、*Ship*1、*Ship*2 というそれぞれの船が *Port* という港に入港しようとしていることを表している。(5) 式から到達可能な (6) 式と (7) 式は互いに構造合同であるため 1 つのノードとして考える。

$$Minato[\ CY[\ ]\ \mid Ship1[\ ]\ \mid Ship2[\ ]\ ] \quad (6)$$

$$Minato[\ CY[\ ]\ \mid Ship2[\ ]\ \mid Ship1[\ ]\ ] \quad (7)$$

次に *stuttering equivalent* に基づく簡約化を考える。プロセス式生成システムで生成されたプロセス式は、「すべてのコンテナを積み込んでから船が出港する」等の動作を制御するための制御用 ambient を数多く持っている。そこで (8) 式のような場合を考える。

$$Port[\ CY[\ Co[\ open\ arrivePort]\ ]\ \mid Ship1[\ arrivePort[\ out\ Ship1\ .\ in\ CY\ .\ in\ Co \mid out\ CY.\ in\ Ship1]\ ]\ ] \quad (8)$$

(8) 式は船が港に入港し、それを確認したうえでコンテナをコンテナ船に積み込むという動作を表している。船の入港を確認し、コンテナの積み込み作業に入るように制御する制御用の ambient は *arrivePort* になる。*arrivePort* の遷移を行うと (9) 式のようにコンテナの移動を行えるようになる。

$$\rightarrow Minato[\ CY[\ Co[\ out\ CY.\ in\ Ship1]\ ]\ ]\ \mid Ship1[\ ]\ ] \quad (9)$$

(8) 式から (9) 式までの遷移を物を表現する ambient に注目してみると階層構造に変化がない。これは *stuttering equivalent* に基づき等価なノードだとみなすことができる。図 2 のように、(9) 式までの 3 回の遷移についても同じことが言えるので、これら 5 つのノードを 1 つにまとめることで遷移グラフの簡約化を行うことができる。

このようにして簡約化された遷移グラフに対して、4 節で示した公理系に基づくモデル検査を行う。

5.2 検証実験

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元に検証実験を行った。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。

$$P \vDash \Box\Diamond\Diamond Port\_B[CY[Co[T]|T]|T]] \quad (10)$$

(10) 式の $CY[Co[T]|T]$ 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 $Co[T]|T$ でコンテナヤード CY の中にコンテナ Co 以外が存在してもよいことを示している。$\Diamond Port\_B[CY[Co[T]|T]|T]]$ は、プロセス式のどこかで $Port\_B[CY[Co[T]|T]|T]]$ 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 (10) 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。

$$\begin{aligned} P \vDash \Box(&\neg\Diamond Port\_A[Ship[T]|T] \\ &\wedge\neg\Diamond Port\_B[Ship[T]|T] \\ &\Rightarrow \neg\Diamond(Ship[T]|Co[T])) \end{aligned} \quad (11)$$

p2 の性質は、コンテナを積み込む前または積み下した後は $(Ship[T] \mid Co[T])$ で表現でき、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 (11) 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。

$$\begin{aligned} P \vDash \Box(&\neg\Diamond Port\_A[\Diamond Co[T]] \\ &\vee\neg\Diamond Port\_B[\Diamond Co[T]] \\ &\Rightarrow \Diamond(Ship[Co[T]|T])) \end{aligned} \quad (12)$$

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ればよいので (12) 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。

- 常に $Port\_A$ は SEA（海）に存在する。（プロセス式に含まれる各港ごとにこの性質を満たす）

$$P \vDash \Box(SEA[Port\_A[T]|T]) \quad (13)$$

- 常に $Port\_A$ にはそれぞれ CY（コンテナヤード）が存在する。（プロセス式に含まれる各港ごとにこの性質を満たす）

$$P \vDash \Box\Diamond(Port\_A[CY[T]|T]) \quad (14)$$

- $Ship$ は SEA（海）の上か $Port\_A$（航路上の港）に存在する。

$$\begin{aligned} P \vDash \Box(&SEA[Ship[T]|T] \vee \Diamond Port\_A[Ship[T]|T] \\ &\vee\Diamond Port\_B[Ship[T]|T] \\ &\vee\Diamond Port\_C[Ship[T]|T]) \end{aligned} \quad (15)$$

- 常に Co は存在する。

$$P \vDash \Box(\Diamond Co[T]|T) \quad (16)$$

- コンテナに別のコンテナは入らない。

$$P \vDash \neg\Box(\Diamond Co1[Co[T]|T]|T) \quad (17)$$

$$P \vDash \neg\Box(\Diamond Co[Co1[T]|T]|T) \quad (18)$$

物理的に起こり得ない性質である s1 は、海、港は (13) 式、コンテナヤードは (14) 式、コンテナ船は (15) 式、コンテナは (16) 式でそれぞれ常に存在することを表現できる。(13) 式、(14) 式、で階層構造が常に固定されており、かつ (13) 式、で海とパラレルに存在するものはないと表現しているので、s2 の性質が言える。s3 は、コンテナ船が、海上か、輸出港、輸入港、経由港のいずれかに存在するということなので (15) 式で表現できる。コンテナが複数在る場合には、(18) 式でそれぞれコンテナに入らないので s4 の性質が言える。

送り状と B/L Instructions の 2 つの書類からプロセス式生成システムで、21 個の ambient を持つプロセス式が生成され、このプロセス式の検証を行った。(10) 式〜(18) 式の論理積を、プロセス式が満たすかどうかの検査に 1 分を要した。OS:Windows XP,CPU:Core2Duo 3.00GHz, メモリ:2.00GB. この程度のプロセス式に対して 1 分ほどかかっているので、今後より効率的な検証を行えるようにする必要がある。

6 結論

本論文では、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を構築した。これによりプロセス式生成システムにより生成されたプロセス式の正当性を示すことができた。しかし、現状ではコンテナ数が 1000 を超えるようなコンテナ輸送に対する検証を行うことができない、今後遷移グラフの縮小や検証の効率化を図り対応する必要がある。

参考文献

[1] 森本大輔: 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).
[2] Luca Cardelli and Andrew D.Gordon: Mobile Ambients, LNCS, Vol.1872, pp.1-38(2000).
[3] Kato,T.et al: The Implementation of Ambient Calculus with HORB for Mobile Agents, Proceedings of The 7th World Multiconference on Systemics, Cybernetics and Infomatics, Vol.2(2003).
[4] Luca Cardelli and Andrew D.Gordon: Ambient Logic, Mathematical Structures in Computer Science(2003).
[5] E.M.Clarke, Orna Grumberg, Doron Peled: Model Checking, Mit Pr(2000).