A Handling Management System for Freight with the Ambient Calculus

Toru Kato, Masahiro Higuchi

***Abstract*—**This paper proposes a freight management system that ensures the correctness of container handling during shipping. The system determines the correctness by comparing container handling, which is sensed by IC tags, with formal models (formulae) written in the ambient calculus. The ambient calculus is a formal description language that is suitable for expressing freight systems with nested structures that dynamically change. The management system generates formulae automatically from several documents used in real freight systems. An implementation of the system and the results of a simple experiment using it are presented.

***Keywords*—**Ambient Calculus, Formal Model, Freight System, Logistics, RFID

I. Introduction

IN the field of distribution, the increase in cargo handling errors is a serious problem as the amount of freight circulation increases. To deal with this problem, several systems using IC tags have been developed to manage distribution. For example, [1] shows an experiment by the Japanese government that used GPS to trace the land routes of trucks carrying luggage with IC tags. Reference [2] details a cooperative experiment by the Japanese and US governments that tried to trace the location of 100 containers with the Marine Asset Tag Tracking System (MATTS) during navigation from Yokohama Port to the Port of Los Angeles.

The information that those systems try to track is, however, restricted to the location of cargo. We focused on the nested structure of freight systems, that is, packages (e.g. containers) containing smaller packages (e.g. luggage) that are accommodated by a larger entity such as a container ship (Fig. 1). Using information about this type of nested structure, we are able not only to trace the route of containers but also to recognize their movement more precisely, including their loading (or unloading) onto (or from) ships.

We are researching a distribution management system based on descriptions of freight systems in the ambient calculus. The ambient calculus [3] (AC) is a formal description language originally developed for describing the nested structure of the Internet or the behavior of mobile agents whose nested structure dynamically changes. The ambient calculus is suitable for modeling freight systems, which also have a dynamically changing, nested structure.

Manuscript received June 30, 2009.Toru Kato and Masahiro Higuchi are with the Department of Informatics, Kinki University, Kowakae 3-4-1, Higashiosaka, Osaka, Japan (phone: +81–6–6730–5880; fax: +81–6–6727–2024; e–mail: kato@info.kindai.ac.jp).

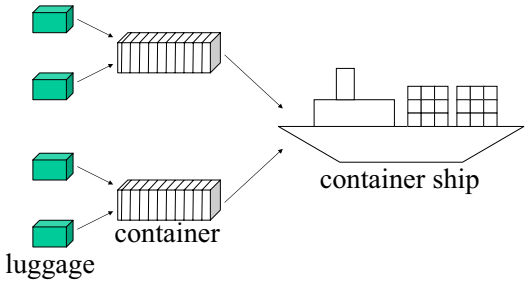

Fig. 1. Nested structure of a freight system

Modeling logistics have been investigated for many purposes, such as estimating the economic effect of constructing railway logistics [4] or dry harbors [5] and for distribution optimization [6]. When logistics for railways or ports are developed, many factors influence decisions about the choices of routes or locations. Simulations of the logistics based on the formal model proposed in [4] and [5] can support such decisions. Reference [6] proposed a way of optimizing distributed logistics based on a multi–agent system for supply chain management.

A way of modeling freight systems is proposed in this paper. The purpose of the model is not to simulate or optimize the freight system, but to serve as a basis for constructing a handling management system. Currently the correctness of the handling of containers in freights depends largely on human labor, though the increase of the amount of freight has made handling more difficult. Our handling management system confirms the movement of containers automatically.

The implementation of the system consists of two components: a system that automatically generates AC formulae and another system that manages the actual freight. The former system generates formulae that express the transportation routes of containers based on several trading documents such as shipping invoices, B/L instructions, container packing lists, and routing plan tables of ships. The latter system senses the movement of real objects such as ships or containers and checks their consistency with the corresponding formulae. We also present the results of a simple

experiment to show the effectiveness of these systems.

This paper is organized as follows. Section II shows the freight systems we concentrated on and the documents used in actual trade. In Section III, we review the syntax and the operational semantics of AC. Section IV explains how freight systems are expressed by AC formulae. In section V, we explain the configuration of our freight system. The results of an experiment on the system are given in Section VI.

II. Freight Systems and Documents

In freight systems called full container load (FCL), each container is occupied by the luggage of one owner. The FCL procedure is as follows. Containers that have already been packed by each sender wait in the container yard of a port to be loaded. After the arrival of the container ship, the containers are brought from the container yard to the ship. The ship leaves the port for its destination port after all its containers are loaded. Several hundred to 8,000 containers (40-foot type) can be loaded on one ship. The containers are unloaded and carried to a container yard after the ship arrives at its destination port. The ship leaves that port after the containers are unloaded there.

The movement of all the containers and the shipping vessel for a shipment are specified by several documents such as shipping invoices, B/L instructions, container packing lists, and routing plan tables of the ship. Some examples of these are shown in Figs. 2 and 3.

Means of Transport & Route		
Shipped p	ShipA	On or About
From	TOKYO , JAPAN	Via
To	KOBE , JAPAN	

Fig. 2. Example of shipping invoice

PRE-CARRIAGE BY		PLACE OF RECEIPT (SERVICE TYPE)
		TOKYO CY
OCEAN VESSEL	VOY.NO.	PORT OF LOADING
SHIP	v1	TOKYO, JAPAN
PORT OF DISCHARGE		PLACE OF DELIVERY (SERVICE TYPE)
KOBE, JAPAN		KOBE CY

Fig. 3. Example of B/L Instruction

The purpose of the management system presented is to automatically check whether containers are carried accurately in accordance with these documents during transportation from a departure container yard to a destination container yard. To do so, we generate a formal model of the freight system from those documents so that machine checking is possible.

III. Ambient Calculus

This section reviews the syntax and the semantics of AC originally defined in [3]. We assume there are infinite sets of *names* ranged over by *n*. *Capabilities* and *processes* are ranged over by *M, N* and *P, Q*, respectively.

Definition 3.1 (Syntax):

n		names	P, Q	$::=$	processes
M, N	$::=$	capabilities	0		inactivity
in n		can enter *n*	$P\|Q$		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			$\square$

The capability "*in* n" means the ambient that encloses it can enter the ambient "n[]". The capability "*out* n" means the ambient that encloses it can exit the ambient "n[]". The capability "*open* n" dissolves the ambient "*n*[]". These actions are prescribed in the reduction rules in Definition 3.2.

Definition 3.2 (Reduction: $\rightarrow$):

$$n[\textit{in } m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$$
$$m[n[\textit{out } m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]$$
$$\textit{open } n.P \mid n[Q] \rightarrow P \mid Q$$

$$P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$$
$$P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$$
$$P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$$
$$P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q' \qquad \square$$

In the first rule of Definition 3.2, the capability "*in* m" leads the ambient "n[]" into the ambient "m[]" and the capability is consumed.

This reduction happens when there are ambient "n[]" and ambient "m[]" in parallel positions, otherwise, ambient "n[]" waits for ambient "m[]" to appear in such a position. This rule enables us to express synchronous actions of objects in a simple way such as that containers are loaded just after a ship arrives at a port.

The syntax of AC originally defined in [3] has x *variables*, (x) (*input*), $\langle M \rangle$ (*output*) and $!P$ (*replication*) that are omitted in Definitions 3.1 and 3.2 because we do not use them for representing the behavior of freight systems in this paper.

IV. Modeling Freight Systems by AC

Formulae of AC that represent the behavior of freight systems are automatically generated by the system from Excel files of the types of trading documents shown in Section II.

These formulae represent the current state of the freight system and the possible future behavior of

objects (ships, ports, container yards, and containers). The formulae are composed of two kinds of ambients: *physical ambients* and *control ambients*. The former represent real objects such as ships, ports, container yards, or containers and the latter are used for controlling the movement of physical ambients. In our system, the reduction concerning a control ambient occurs immediately when it is enabled whereas that of a physical ambient occurs when it is enabled and movement of the corresponding real object is sensed.

This section shows how freight systems are modeled by these ambients and explains the role of control ambients.

A. Ship Ambient

We define the general form of a ship ambient as (1).

$$\begin{aligned} &\textit{SHIP\_}vn[\ \textit{in PORT1.open cnt.out PORT1} \\ &\qquad \textit{.in PORT2.open ucnt.out PORT2} \\ &\qquad |\textit{open lcomp\_}am\textit{.cnt}[\ ] \\ &\qquad |\textit{open ulcomp\_}am\textit{.ucnt}[\ ]]. \end{aligned} \tag{1}$$

The ambients with capitalized names are physical ambients and the others are control ambients. "*PORT1*" is the place holder for the name of a loading port and "*PORT2*" is for that of an unloading port. *SHIP*, vn, and am are the place holders for the name of a ship, the voyage number of a ship, and the container's ID, respectively.

We use (1) as a template for generating a ship ambient. All place holders are replaced with the real names written in the trading documents, and the number of capabilities "*open cnt*", "*open ucnt*","*open lcomp_*am", and "*open ulcomp_*am" and the number of control ambients "*cnt*[]" and "*ucnt*[]" are decided according to the number of containers loaded onto the ship in a real shipment. The name "(*u*)*lcomp*" represents the completion of (un)loading a container. An example of a ship ambient generated using the template is (6) shown in IV-E.

The ship actions we are concerned with are entering a port, loading and unloading containers, and leaving the port. The action of entering a port is simply expressed by the enter capability "*in PORT1*" in (1).

The action of loading containers is controlled by the capabilities of container ambients, explained in IV-C.

A ship must not leave a port until all the containers that must be loaded at the port are loaded, thus, we need an elaborate formula for expressing the action of leaving a port as follows. The "*out PORT1*" capability in (1) is blocked by the preceding capability "*open cnt*". The control ambient "*cnt*[]" is also preceded by the "*open lcomp_am*" capability that will be consumed after the container ambient "*CO_am*[]" comes into the "*SHIP_*vn[]" ambient and the control ambient "*lcomp_am*[]" appears in the "*SHIP_*vn[]" ambient (see IV-C).

Similarly, the capabilities "*open ucnt*", "*open ulcomp_am*", and the control ambient "*ucnt*[]" in (1) are used to prevent the "*SHIP_*vn[] ambient from leaving the ambient "*PORT2*" before unloading the container.

B. Port and Container Yard Ambient

The template for a loading port ambient is (2).

$$\begin{aligned} &\textit{PORT1}[\\ &\qquad \textit{load\_}vn[\textit{in SHIP\_}vn\textit{.out SHIP\_}vn\textit{.in CY}] \\ &\qquad |CY[]]. \end{aligned} \tag{2}$$

A loading port ambient contains a control ambient and the container yard ambient "*CY*[]". The control ambient "*load_*vn[]" waits for the "*SHIP_*vn[]" ambient to enter the port ambient, and after confirming it, the control ambient enters the "*CY*[]" ambient in order to notify container ambients in the container yard ambient that the ship ambient entered the port ambient.

The template for an unloading port ambient is (3).

$$\begin{aligned} &\textit{PORT2}[\textit{uload\_}vn_{\textit{PORT2}}[\textit{in SHIP\_}vn] \\ &\qquad |\textit{CY}[\textit{ulcomp\_}am[\textit{in CO\_}am\textit{.out CO\_}am \\ &\qquad\qquad \textit{.out CY.in SHIP\_}vn]]]. \end{aligned} \tag{3}$$

When the "*SHIP_*vn[]" ambient enters the port ambient, the control ambient "*unload_vn_PORT2*[]" enters the ship ambient to notify container ambients that the ship ambient entered the unloading port ambient. After the notification, container ambients are ready to exit the ship ambient and enter the container yard ambient.

As soon as the container ambient "*CO_am*[]" enters the container yard ambient, the control ambient "*ulcomp_am*[]" enters the ship ambient to notify the ship ambient that the container ambient entered the container yard ambient. After the notification, the ship ambient is ready to exit the port ambient because the capability "*open ucnt*" in (1) that blocks the "*out PORT2*" capability is consumed.

C. Container Ambient

The template for a container ambient is (4). When a container ambient is generated using the template (4), it is placed in a container yard ambient generated using (2).

$$\begin{aligned} &\textit{CO\_}am[\textit{in load\_}vn\textit{.out load\_}vn\textit{.out CY} \\ &\qquad \textit{.in SHIP\_}vn\textit{.lcomp\_}am[\textit{out CO\_}am] \\ &\qquad |\textit{in uload\_vn\_PORT2.out uload\_vn\_PORT2} \\ &\qquad \textit{.out SHIP\_}vn\textit{.in CY} \\ &\qquad \textit{.ulcomp\_}am[\textit{out CO\_}am\textit{.out CY} \\ &\qquad\qquad \textit{.in SHIP\_}vn]]. \end{aligned} \tag{4}$$

The capabilities "*in load_*vn*.out load_*vn" are consumed when the control ambient "*load_*vn[]" in (2) appears in the ambient "CY[]" in (2), then the capabilities "*out CY.in SHIP_*vn" become executable. This means when the ship arrives at the port, containers are

allowed to be brought out from the container yard to the ship.

As soon as the container ambient enters the ship ambient, the control ambient "*lcomp_am*[]" exits the container ambient and appears in the ship ambient. These reductions express the notification to the ship that the container has been loaded onto the ship. Then, as explained in IV-A, the capability "*out PORT1*" of the ship ambient in (1) becomes executable.

The "*out SHIP_vn*" capability is blocked by the capabilities "in *uload_vn_PORT2.out uload_vn_PORT2*" that are consumed after the ship ambient enters an unloading port ambient and the control ambient "*uload_vn_PORT2*[]" in (3) enters the container ambient. Then the capability "*out SHIP_vn.in CY*" of the container ambient becomes executable. This means when the ship arrives at the unloading port, containers are allowed to be brought out from the ship and into the container yard of the port.

As soon as the container ambient enters the container yard ambient, the control ambient "*ulcomp_am*[]" in (4) exits the container ambient and container yard ambient and enters the ship ambient. These reductions express the notification from the port to the ship that the unloading of the container has been completed.

When the control ambient enters the ship ambient, the capabilities "*ulcomp_am*" and "*ucnt*" in (1) are consumed and the capability "*out PORT2*" of the ship becomes executable. This means the ship is allowed to leave the unloading port after the unloading of the container has been completed.

D. SEA Ambient

The template for a sea ambient is (5). When a handling checking starts, the sea ambient generated using the template (5) is placed in the PC of a port to check the movement of the ship.

$$SEA[PORT1[\ ]\ |SHIP\_vn[\ ]\ |\ PORT2[\ ]]. \tag{5}$$

E. Examples of Ambients

We give examples below of formulae generated by the system to represent a shipment of 500 containers from Tokyo Port to Kobe Port.

$$\begin{aligned}
&SHIP_v1[in\ TOKYO\\
&\quad .open\ cnt.\cdots .open\ cnt\\
&\quad .out\ TOKYO.in\ KOBE\\
&\quad .open\ ucnt.\cdots .open\ ucnt.out\ KOBE\\
&\quad |open\ lcomp_a1.cnt[\ ]\\
&\qquad |\cdots|open\ lcomp_a500.cnt[\ ]\\
&\quad |open\ ulcomp_a1.ucnt[\ ]\\
&\qquad |\cdots|open\ ulcomp_a500.ucnt[\ ]].
\end{aligned} \tag{6}$$

$$\begin{aligned}
&TOKYO[\\
&\ load_v1[in\ SHIP_v1.out\ SHIP_v1.in\ CY]\\
&|CY[\\
&\ CO_a1[in\ load_v1.out\ load_v1.out\ CY\\
&\qquad .in\ SHIP_v1.lcomp_a1[out\ CO_a1]\\
&\qquad |in\ uload_v1_KOBE.out\ uload_v1_KOBE\\
&\qquad .out\ SHIP_v1.in\ CY\\
&\qquad .ulcomp_a1[out\ CO_a1.out\ CY\\
&\qquad\qquad .in\ SHIP_v1]]\\
&\ |\ \cdots\ |\ CO_a500[\cdots]]].
\end{aligned} \tag{7}$$

$$\begin{aligned}
&KOBE[uload_v1_KOBE[in\ SHIP_v1]\\
&\qquad |CY[ulcomp_a1[in\ CO_a1.out\ CO_a1\\
&\qquad\qquad .out\ CY.in\ SHIP_v1]\\
&\qquad\quad |\cdots|\ ulcomp_a500[\cdots]]].
\end{aligned} \tag{8}$$

Formula (6) represents a ship that will enter Tokyo Port, load containers there, and carry them to Kobe Port. The formula (7) represents Tokyo Port and its container yard where 500 containers $CO\_a1\cdots CO\_a500$ wait to be loaded. Formula (8) represents Kobe Port. As described in IV-D, all of the formulae are in the ambient "*SEA*[ ]".

F. Handling Homogeneity of Container Ambients

We could simplify (1) as (9).

$$\begin{aligned}
&SHIP_vn[\ in\ PORT1.open\ lcomp_am\\
&\qquad .out\ PORT1.in\ PORT2\\
&\qquad .open\ ulcomp_am.out\ PORT2].
\end{aligned} \tag{9}$$

Then, the ship ambient generated using the template (9) that expresses a ship loading 500 containers would be simpler than (6) as (10).

$$\begin{aligned}
&SHIP_v1[in\ TOKYO\\
&\quad .open\ lcomp_a1.\cdots .open\ lcomp_a500\\
&\quad .out\ TOKYO.in\ KOBE\\
&\quad .open\ ulcomp_a1.\cdots .open\ ulcomp_a500\\
&\quad .out\ KOBE].
\end{aligned} \tag{10}$$

The formula (10) is based on the assumption that containers are loaded in order of their container IDs, though in real freight systems, container loading does not necessarily follow ID order. The homogeneous expression concerning containers in (6) enables us to check the handling of container loading in any order. Besides, homogeneity of container ambients in (6) is inevitable for a partial order reduction in the model checking of freight systems.

V. Freight Management System

The freight management system maintains the formulae representing the current state of the freight system. The system determines whether the container handling is valid by sensing the movement of containers by RFID and checking whether the reductions representing the movement are possible in the current formulae.

A. AC processing system

The AC processing system [7] reduces processes (formulae) according to Definition 3.2 on distributed environments. Each processing system contains several ambients (a part of the entire formula) and refers remote ambients if necessary.

In the processing system, physical ambients and control ambients are reduced in different ways as follows.

Physical ambients: when the processing system is notified of the movement of a real object (ship or container), the system checks whether the corresponding reduction is enabled for the formula representing the object. If it is enabled, the system confirms that the movement is valid and reduces the formula. If not, the system issues a warning.

Control ambients: the reductions related to control ambients are made in the processing system as soon as they are enabled.

B. System Configuration

The devices used in our system are shown in Fig. 4. We will set up a PC with an AC processing system in

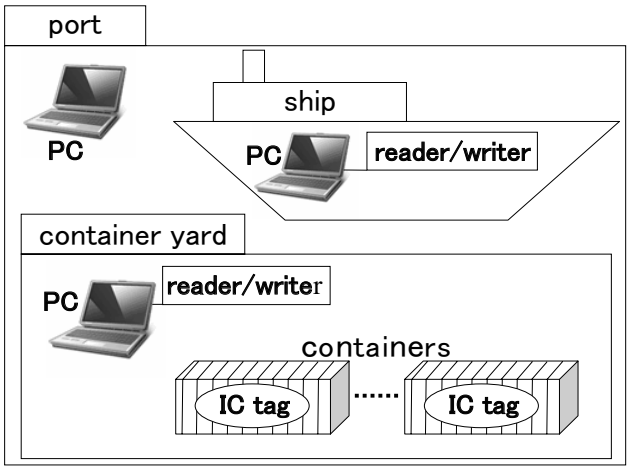

Fig. 4. System configuration

the office of each port, in each container yard, and in the cargo room of each ship. RFID reader/writers are connected to the PCs in the container yards and the ship cargo rooms. An IC tag is put on each container.

We put the formulae generated from the trading documents into the appropriate PCs and IC tags. The formulae are decomposed and distributed into the PCs and IC tags. When, for example, containers are shipped from Tokyo Port to Kobe Port, the formula (7) is decomposed into the formulae (11), (12), and 500 formulae like (13). Then they are distributed to the PC in Tokyo Port, the PC in the container yard of Tokyo Port, and the IC tags of the containers.

$$\begin{aligned} &\mathit{TOKYO}[\\ &\quad \mathit{load\_v1}[\mathit{in\ SHIP\_v1.out\ SHIP\_v1.in\ CY}]\\ &\quad |\mathit{CY}[\,]]. \end{aligned} \tag{11}$$

$$\mathit{CY}[\mathit{CO\_a1}[\,] \mid \cdots \mid \mathit{CO\_a500}[\,]]. \tag{12}$$

$$\begin{aligned} &\mathit{CO\_a1}[\mathit{in\ load\_v1.out\ load\_v1.out\ CY}\\ &\qquad \mathit{.in\ SHIP\_v1.lcomp\_a1}[\mathit{out\ CO\_a1}]\\ &\qquad |\mathit{in\ uload\_v1\_KOBE.out\ uload\_v1\_KOBE}\\ &\qquad \mathit{.out\ SHIP\_v1.in\ CY}\\ &\qquad \mathit{.ulcomp\_a1}[\mathit{out\ CO\_a1.out\ CY}\\ &\qquad\qquad \mathit{.in\ SHIP\_v1}]]. \end{aligned} \tag{13}$$

Similarly, formula (8) is decomposed to (14) and (15).

$$\mathit{KOBE}[\ \mathit{uload\_v1\_KOBE}[\mathit{in\ SHIP\_v1}]|\mathit{CY}[\,]]. \tag{14}$$

$$\begin{aligned} &\mathit{CY}[\ \mathit{ulcomp\_a1}[\mathit{in\ CO\_a1.out\ CO\_a1}\\ &\qquad \mathit{.out\ CY.in\ SHIP\_v1}]\\ &\quad |\cdots|\ \mathit{ulcomp\_a500}[\cdots]]. \end{aligned} \tag{15}$$

This means each of the processing systems in each PC maintains a physical ambient, i.e., maintains all the control ambients and the names of all the physical ambients in it. Distributed management like this enables us to reduce the amount of traffic between PCs necessary for managing freight systems.

The physical ambient for the sea that expresses the current situation of the sea must be maintained in a PC and we assign that role to the PC of a port. Thus, the PC of the port has two AC processing systems, one is maintaining the port ambient (11), and the other is maintaining the sea ambient (16) that is generated using the template (5). We call the former the processing system for the sea and the latter the processing system for Tokyo port.

$$\mathit{SEA}[\mathit{TOKYO}[\,]\ |\mathit{SHIP\_v1}[\,] \mid \mathit{KOBE}[\,]]. \tag{16}$$

When we need to check the locations or the states of all objects, all the PCs communicate and we can make a formula representing the entire state of the freight system.

C. Checking the Container Handling

Whenever a container is brought out from or into a container yard or a ship, the RFID reader/writer there reads the IC tag on the container and the management system on the PC determines if that movement is valid. The system can detect container handling errors and incorrect ship movements as follows.

Errors in handling of containers:

- bringing out containers from a container yard before the arrival of a container ship
- loading containers onto the wrong ship
- unloading containers to the wrong port

Incorrect departure of ships:

- leaving containers behind at a loading port
- leaving containers in the ship at an unloading port

This subsection details how the system checks the handling of bringing containers out from a container yard to a ship.

1) Bringing Out Containers: Figure 5 is an example of the system determining that the handling of a container is not valid when the container *CO_a*1 is brought out from the container yard before the ship arrives at Tokyo port.

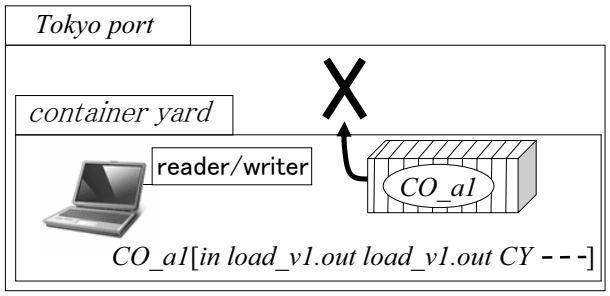

Fig. 5. Checking movement (exiting container yard)

When the container *CO_a*1 passes the gate of the container yard, the RFID reader/writer reads the IC tag and the processing system gets the formula (13). Then the formula in the PC becomes (17).

$$CY[CO\_a1[in\ load\_v1.out\ load\_v1.out\ CY\cdots] \\ |CO\_a2[\,]|\cdots|CO\_a500[\,]]. \quad (17)$$

The processing system finds the corresponding reduction is impossible because the capabilities "*in load_v*1.*out load_v*1" are in front of the capability "*out CY*", so the management system issues a warning. This means the ship *SHIP_v*1 has not arrived and bringing containers out from the container yard has not been permitted yet.

After the ship arrives, the handling management system confirms that containers can be brought out from the container yard as follows. When the ship *SHIP_v*1 is about to enter Tokyo Port, the PC on the ship sends the formula "*SHIP_v*1[*in TOKYO*]" to the PC maintaining the sea ambient. That is communication (a) illustrated in Fig. 6.

As the AC processing system for the sea maintaining (16) recognizes that the ambient "*SHIP_v*1[]" and the ambient "*TOKYO*[]" are in parallel positions in the ambient "*SEA*[]", (16) becomes (18), and (18) is reduced to (19) in the processing system.

$$SEA[TOKYO[\,]\,|SHIP\_v1[in\ TOKYO]\,|\,KOBE[\,]]. \quad (18)$$

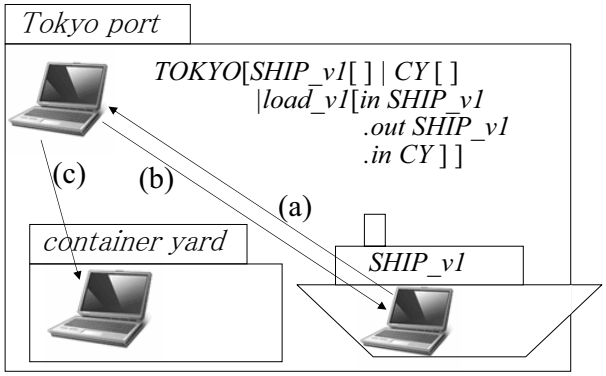

Fig. 6. Communication between PCs

$$SEA[TOKYO[\,]\,|\,KOBE[\,]]. \quad (19)$$

The ambient "*SHIP_v*1[]" disappeared in (19) because the inside of the ambient "*TOKYO*[]" is maintained by the processing system for Tokyo port. Thus, the processing system for the sea sends the ambient "*SHIP_v*1[]" to the the processing system for Tokyo port, and (11) becomes the formula shown in Fig. 6 in the processing system for Tokyo port. Then the PC in the port notifies the PC in the ship of the confirmation. This is communication (b) illustrated in Fig. 6. Then the processing system in the PC of the ship reduces (6) to (20) by consuming the capability "*in TOKYO*".

$$SHIP\_v1[\ open\ cnt.\cdots.open\ cnt.out\ TOKYO \\ .in\ KOBE\cdots(\text{same to}(6)). \quad (20)$$

After several reductions, the formula in the processing system for Tyoko port is reduced to (21).

$$TOKYO[SHIP\_v1[\,]|CY[\ load\_v1[\,]]]. \quad (21)$$

Since the PC in the container yard maintains the inside of the ambient "*CY*[]", the PC at the port sends the ambient "*load_v*1[]" to the PC in the container yard. This is communication (c) illustrated in Fig. 6. After the communication, the formula (12) that is maintained in the PC of the container yard becomes (22).

$$CY[\ load\_v1[\,]|CO\_a1[\,]\ |\ \cdots\ |\ CO\_a500[\,]]. \quad (22)$$

As a result, when a container with the ambient "*CO_a*1[]" passes the gate and the RFID reader/writer reads the IC tag that contains (13), the formula in the PC of the container yard becomes (23).

$$CY[\ load\_v1[\,] \\ |CO\_a1[in\ load\_v1.out\ load\_v1.out\ CY\cdots] \\ |CO\_a2[\,]|\cdots|CO\_a500[\,]]. \quad (23)$$

Now, the handling management system confirms the container with the ambient "*CO_a*1[]" should be brought out from the container yard because the capabilities "*in load_v*1.*out load_v*1" are enabled and

consumed immediately and (23) is reduced to (24), in which the capability "*out CY*" is enabled.

$$CY[\ load\_v1[\ ]|CO\_a1[out\ CY\cdots] \\ |CO\_a2[\ ]|\cdots|CO\_a500[\ ]]. \quad (24)$$

When the container is brought out from the container yard, the RFID reader/writer writes the formula (25) back to the IC tag of the container, which is the remains of (13) after those reductions.

$$CO\_a1[in\ SHIP\_v1.lcomp\_a1[out\ CO\_a1] \\ |in\ uload\_v1\_KOBE.out\ uload\_v1\_KOBE \\ .out\ SHIP\_v1.in\ CY \\ .ulcomp\_a1[out\ CO\_a1.out\ CY \\ .in\ SHIP\_v1]]. \quad (25)$$

2) Container Entering: Figure 7 shows a situation where the container with the ambient "*CO_a1*[]" is about to be loaded into a ship with the ambient "*SHIP_v1*[]".

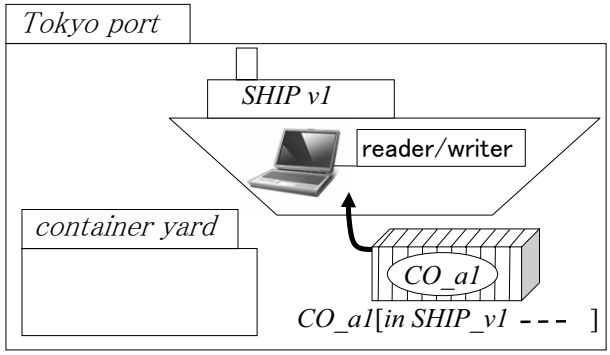

Fig. 7. Checking movement (enter ship)

When the container CO_a1 passes the entrance of the ship, the RFID reader/writer reads the IC tag on the container and the processing system in the PC of the ship gets formula (25). Then the formula on the PC on the ship becomes (26).

$$SHIP\_v1[\cdots]|CO\_a1[in\ SHIP\_v1\cdots]. \quad (26)$$

Since formula (26) can be reduced to (27), the management system confirms the loading of the container.

$$SHIP\_v1[\cdots|CO\_a1[\cdots]]. \quad (27)$$

After the reduction, the RFID reader/writer writes the ambient "$CO\_a1[\cdots]$" in formula (27) back to the IC tag of the container.

When a container is about to be loaded onto the wrong ship, for example a ship with the ambient "*SHIP_v2*[]", the PC on the ship has formula (28) that cannot be reduced any more, and the management system rejects the loading by showing a warning on the PC monitor.

$$SHIP\_v2[\cdots]|CO\_a1[in\ SHIP\_v1\cdots]. \quad (28)$$

D. Programs and Devices

All programs are written in Java and the sizes of the programs are as follows.

AC formulae generating system: 24 Java Classes, 3.3 K steps

Container handling management system: 72 Java Classes, 14.6 K steps

GUI: 28 Java Classes, 6 K steps

We use the following RFID devices.

RFID reader/writer: SkyeModule M1 (SkyeTek Co., Ltd) using 13.56 MHz, 50 cm communication distance.

IC tag: RF-T5M40 (Keyence Co., Ltd) with 1 Kb memory.

VI. Experiment

This section describes a simple experiment simulating the transportation of five containers from Tokyo Port to Kobe Port on one container ship.

First, we confirmed that the formula-generating system generated the formulae (6), (11), (12), (14), (15), (16) and five formulae like (13) for containers based on the documents related to the shipping of five containers from Tokyo Port to Kobe Port. (We also confirmed that the system was able to generate formulae for more than 1,000 containers). We wrote each container formula

Fig. 8. Sensing an IC tag with a RFID reader/writer

(13) onto an IC tag that was put on a box representing a container (Fig. 8) and deployed the formulae for the ship (6), Tokyo Port (11) and its container yards (12), Kobe Port (14) and its container yards (15) and the sea (16) on their respective PCs.

Then we started checking. As shown in Fig. 8, we moved the boxes (containers) by hand and made sure the RFID reader/writers could read the formulae, the processing system reduced those formulae, the PCs sent the formulae, the management system determined the validity of the movement, and the RFID reader/writers wrote the formulae back onto the IC tags after the reductions. In Fig. 9, the window of the GUI system shows the contents of all formulae just after the experiment started. Each rectangle in the window represents a

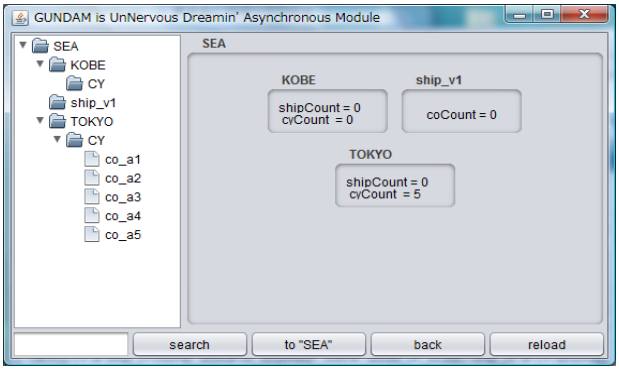

Fig. 9. Visual information for each ambient

real object such as a port or a ship. Clicking a rectangle brings up a display of more precise information.

When we moved a container correctly, it took 7.8 seconds for the system to read the formula (13) from the IC tag, confirm the movement, reduce the formula to formula (25), and write the formula (25) back onto the IC tag.

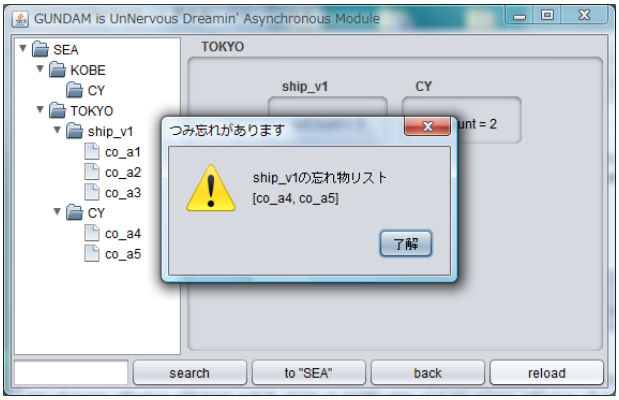

Fig. 10. Warning of a mistake

A warning dialog box appeared when we made the ship leave the port before the completion of container loading (Fig. 10). The Japanese messages in the dialog read "you left something behind", "list of the things left behind [*CO*_a4, *CO*_a5]", and "OK?".

VII. Conclusions

The freight management system presented in this paper confirms whether containers are handled correctly during shipping by comparing the handling to formal AC formulae. The formulae are generated based on documents used in actual trading, so, when the system confirms the correctness of the treatment of the containers, it means that the containers have been handled in accordance with the instructions in the documents. We confirmed that the system determined the validity of the container handling in a practical amount of time, so we think will be able to test the system at a practical scale by using more efficient UHF RFID devices.

In order to apply our system to real logistics, we need (i) to be able to deal with a variety of container shipments and (ii) to show the correctness of the formulae generated by AC generating system.

(i) In real maritime logistics, there are many other types of shipping patterns than are presented in this paper. For example, a ship loads containers whose destinations vary, a ship loads containers at several ports, a container leaves a spoke port for another spoke port via more than one hub port, etc. In order to treat different kinds of shipping, we will have to modify the AC processing system.

(ii) As the formulae are written in the ambient calculus, ambient logic model checking is possible. The ambient logic [8], [9] is a kind of temporal logic that can express properties of ambient formulae such that "the ambient *CO*[] will eventually appear in the ambient *PORT2*[]". In order to perform the model checking for the shipping of more than 1,000 containers, we need to solve the state explosion problem by the partial order reduction [10] and more sophisticated formulae than are presented in this paper. Such model checking will enable us not only to ensure the validity of the formulae but also to make sure that the containers will be delivered to their destinations correctly in any case before the transportation.

References

[1] Ministory of Land Infrastructure Transport and Tourism, "Demonstration driving experiment to develop a L and R route in mekong Region," *http://www.meti.go.jp/english/newtopics/data/nBackIssue20071018_09.html*, 2007.

[2] Ministory of Land Infrastructure Transport and Tourism, "Marine Asset Tag Tracking System (in Japanese)," *http://www.mlit.go.jp/report/press/port02_hh_000006.html*, 2008.

[3] L. Cardelli and A. D. Gordon, "Mobile ambients," *Theoretical Computer Science*, vol. 240, pp. 177–213, 2000.

[4] C. Caballini, P. P. Puliafito, R. Revetria, and F. Tonelli, "Simulation based design for a railway logistics re-engineering project," *International Journal of Mathematics and Computers in Simulation*, vol. 2, pp. 195–205, 2008.

[5] A. D. Marco and C. Rafele, "System dynamics simulation: an application to regional logistics policy making," *International Journal of Mathematics and Methods in Applied Sciences*, vol. 1, pp. 253–260, 2007.

[6] N. Zoghlami and S. Hammadi, "Organization and optimization of distributed logistics: estimation and patrolling approach based on multi–agent system," *International Journal of Mathematics and Computers in Simulation*, vol. 1, pp. 73–80, 2007.

[7] T. Kato, D. Ikeda, and Y. Okada, "The implementation of ambient calculus with HORB for mobile agents," in *Proc. of The 7th World Multiconference of Systemics, Cybernetics and Informatics*, vol. II, 2003, pp. 367–372.

[8] L. Cardelli and A. D. Gordon, "Any time anywhere modal logics for mobile ambients," in *POPL'00*. Proceedings of the 2000 ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2000, pp. 365–377.

[9] L. Cardelli and A. Gordon, "Deciding validity in a spatial logic for trees," *Journal of Functional Programming*, vol. 15, pp. 543–572, 2005.

[10] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 2000.