A Specification Framework for Earth-friendly Logistics

Ichiro Satoh

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
E-mail: ichiro@nii.ac.jp

Abstract. This paper describes the use of a formal approach to logistics management systems to reduce the environmental impact of logistics operations. Trucks play an essential role as carriers in modern logistics services, but collectively they emit a huge quantity of carbon dioxide. To reduce fossil fuel consumption and carbon dioxide emissions resulting from transport, we must enhance the transport efficiency of trucks. The milk-run approach is one of the most effective and popular solutions to this problem. However, it tends to be too complicated to implement in a logistics management system. The framework described in this paper provides a language for specifying the routes of trucks and an order relation as a route selection mechanism. The former is formulated as process calculus and the latter selects suitable trucks according to their routes. This paper also describes a prototype implementation of the framework as a distributed logistics management system based on the use of RFID tags.

1 Introduction

Most transport logistics operations involve huge numbers of trucks, with each truck consuming large quantities of fossil fuel and discharging a large quantity of carbon dioxide (CO_2) into the atmosphere. To reduce fossil-fuel consumption and CO_2 emissions from transport, we need to enhance the efficiency of trucks. The *milk-run* approach, which is one of the most efficient and popular ways of improving truck-load ratios, refers to a means of transportation in which a single truck cycles around multiple suppliers to collect or deliver freight. The name is derived from the milk-runs carried out by farmers collecting milk from dairy cows spread out over pastures. For example, suppose five suppliers, e.g., dairy farmers, send their products to the processing plant every weekday. Using the milk-run approach, one truck calls at each of the suppliers on a daily basis before delivering the collected milk to the customer's plant. In a more traditional approach, e.g., the *Just-In-Time* approach, all suppliers have their own trucks and send one truckload per day to the customer (Figure 1).

Recently, a variety of industries, e.g., food and automobile manufacturers, in addition to the dairy industry, have attempted to use the milk-run approach to reduce the environmental impact of their logistics operations. However, in the milk-run approach, they have to provide multiple trucks using varied routes to satisfy the needs of customers and cater for the requirements of the products. Therefore, the customers and suppliers are confronted by another problem: they need to design truck routes and select suitable trucks with routes that satisfy their requirements.

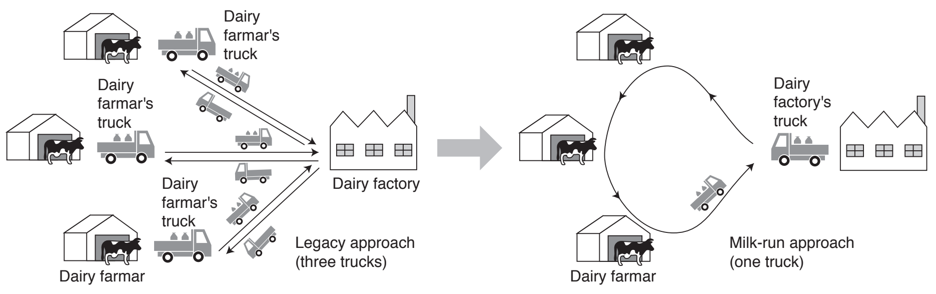

Fig. 1. Legacy approach vs. Milk-run approach.

This paper proposes a novel framework for specifying truck routes and selecting appropriate trucks. The framework introduces a specification language that describes truck routes and a mechanism for selecting suitable milk-run trucks. Since the language is formulated based on an extended process calculus for specifying and reasoning on the routes of trucks, we can determine whether a truck can visit various points, e.g., farmers and manufacturers, along its route to collect or deliver items. The mechanism enables collection/delivery points to select trucks according to the truck route because the route a truck takes is critical in determining its efficiency. The framework was inspired by our experience of real logistics systems. We implemented a prototype of the framework in a distributed logistics management system. We believe that this framework provides a novel and practical application of process calculi in the real world. However, we leave the theoretical aspects of the framework to our future papers because this paper addresses a fundamental platform for managing a milk-run logistics operation.

This paper is organized as follows: Section 2 presents the basic ideas behind the framework. Section 3 defines a process calculus for specifying the routes of trucks for a milk-run logistics-operation, and Section 4 presents an order relation over terms of the languages as a mechanism for selecting routes. Section 5 describes a prototype implementation of the framework and a typical application scenario and Section 6 includes a survey of related work. Section 7 discusses our future work and Section 8 has concluding remarks.

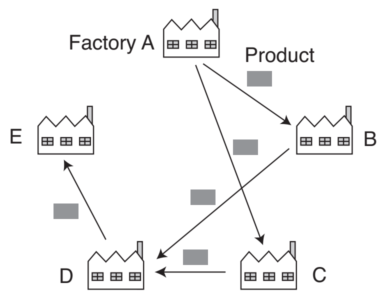

Fig. 2. Five factories with dependencies

2 Background

This paper describes a formal method for specifying the routes of trucks and selecting appropriate trucks to support milk-run operations in transport logistics.

2.1 Example Scenario

Before discussing the framework proposed in this paper, we describe our basic example scenario. Figure 2 shows five factories, A, B, C, D, and E, that have the following dependencies:

- Factory A manufactures products and ships the products to factories B and C.
- Factory B manufactures products and ships the products to factory D.
- Factory C manufactures products and ships the products to factory D.
- Factory D manufactures products and ships the products to factory E.

We assume that a truck has sufficient carrying capacity. It starts at factory A and may visit factory A again. Figure 3 shows four trucks carrying out milk-runs on different routes. The first, second, and third trucks can satisfy the above requirements but the fourth cannot. The third is less efficient than the first and second on their rounds. The framework proposed in this paper was inspired by our real experiences. Although the milk-run approach is effective in reducing the amount of CO_2 emitted by trucks, its management tends to be complicated, which is one of the most significant barriers preventing wider adoption of the approach in real logistics.

2.2 Requirements

This paper assumes that one or more trucks involved in milk-run logistics operations call at multiple points along their routes. Customers and suppliers have to decide which truck and which route will best satisfy their requirements, and this decision is not an easy one. The framework must therefore satisfy the following demands of real logistics systems.

- One or more trucks are available for a milk-run, but their routes may be different. Therefore, points, i.e., suppliers and customers, need to select appropriate trucks according to truck routes. This framework therefore needs to provide a mechanism for selecting truck routes.
- Trucks may be shared by multiple suppliers and customers, so that they collect products at one or more source points and deliver the products at one or more destination points on their way. The trucks need to visit the source points before they visit the destination points. The framework therefore needs to specify the order in which trucks call at various points.
- The routes taken by trucks may also affect product quality. For example, foods should be transported by the shortest route possible to keep their freshness, and perishable foodstuffs should be picked up later than preservable foodstuffs and taken to a food processor or consumer.

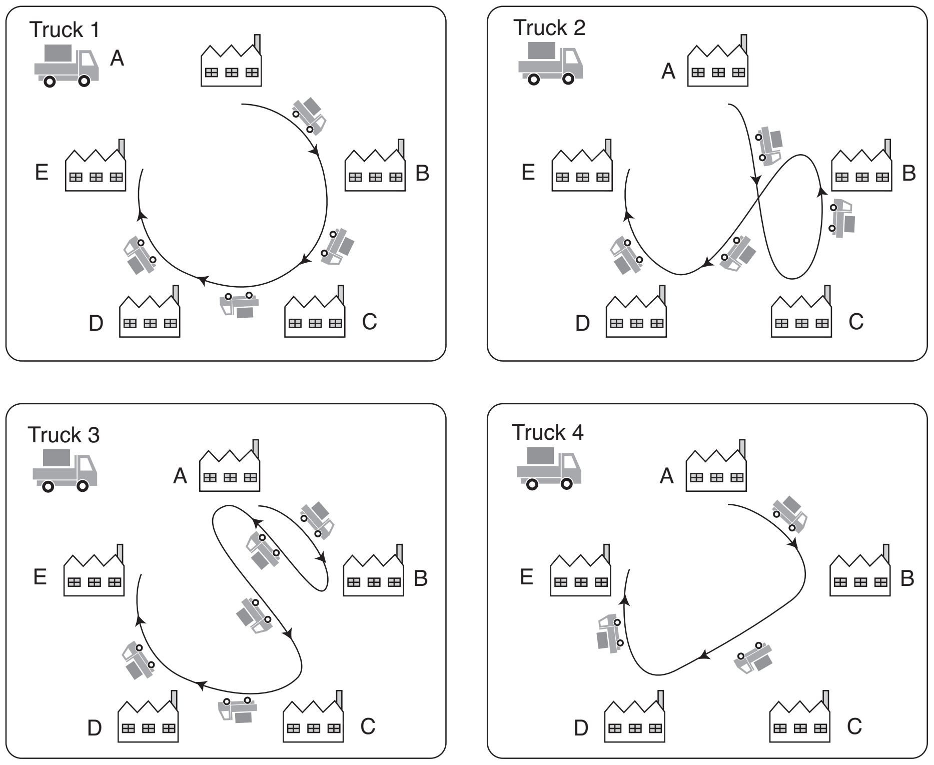

Fig. 3. Four trucks for milk-run operation

– Some products may be collected/delivered at points by trucks without any need for a specific order of arrival at collection/delivery points. That is, the order of the movement of trucks between points does not affect the efficiency of the trucks' operations. Suppliers or customers should select a truck according to the number of movements between the points that the trucks visit.
– Truck routes tend to be regular and static, although they may be changed weekly or monthly. Nevertheless, trucks may bypass some points or take shortcuts without stopping at specified points if they have no freight to deliver to the points or products to pick up.
– Pallets or boxes that contain multiple products are considered as transport units in many current logistics systems, rather than as individual products. These types of containers may have multiple destinations and the receivers may take only some of the products in the container when it arrives at their point.
– In real logistics systems, most points, i.e., suppliers and customers, involve small to medium enterprises or individual operators. They do not want to invest in any additional equipment to support milk-run logistics.
– Current logistics management systems rely heavily on barcodes or RFID tags attached to products or containers. The framework should therefore be compatible

with existing infrastructure and equipment for reading barcodes or tags. However, two-dimensional barcodes or RFID tags do not hold a large amount of data, e.g., up to 128 bytes.[1] The time required to read data from an RFID tag tends to be proportional to the size of the data. The length of route specification should be compact.

2.3 Basic Approach

Truck routes and the requirements of suppliers and customers are various and complex. The selection of trucks for milk-run operations is critical for industrial efficiency and for minimizing carbon dioxide emissions. Careful consideration must be given to selecting suitable trucks with routes that satisfy the requirements of customers and suppliers. Therefore, we need a formal method to solve this problem. To satisfy the above requirements, the framework has two parts.

- It provides a specification language for describing and analyzing truck routes. The language is aimed at specifying only the routes of trucks formulated as an extended process calculus with the expressiveness of truck routes between collection/delivery points.
- It framework defines an algebraic order relation over the terms of the language. The relation is defined based on the notion of bisimulation and compares possible truck routes and the routes required by its specifications. This allows us to accurately determine whether the former satisfies the latter.

Note that the order relation is not intended to generate the most efficient route. Thus, the computational complexity for this relation is not large. Since our goal is to develop a suitable mechanism for selecting trucks for milk-run logistics operations, this paper does not limit the types of products that the trucks carry.

2.4 Remarks

- The framework does not assume any particular logistics system and the specifications of all trucks are independent of any particular logistics service.
- The current implementation does not support any real-time constraints. However, the timing of a truck's arrival at various points tends to depend on external factors, e.g., traffic congestion and the cost of the shipment, in real logistics systems. The milk-run approach is, by its nature, suitable for earth-friendly logistics systems, but not for just-in-time ones.
- The framework is not intended to provide route optimization because truck routes tend to be designed according to external factors.
- Some readers may think that simple executable languages, such as Lisp and Prolog, should be used to specify routes, but it is difficult to verify whether or not routes written in such languages will satisfy the requirements of customers and suppliers because these languages have many primitives that are not used in describing routes.

[1] The amount of data held by barcodes and RFID tags depends on individual systems.

3 Specification Language for Milk-Run Truck Routes

This section defines a language for specifying and reasoning about truck routes. The language consists of two classes. The first is designed to specify truck routes and the second is designed to specify the routes required by products or customers.

Definition 1 The set $\mathcal{E}$ of expressions of the language, ranged over by $E, E_1, E_2, \ldots$ is defined recursively by the following abstract syntax:

$$\begin{array}{rl} E ::= \mathtt{0} & \mid\ \ell \ \mid\ E_1 \mathbin{\mathtt{;}} E_2 \ \mid\ E_1 + E_2 \\ & \mid\ E_1 \mathbin{\#} E_2 \ \mid\ E_1 \mathbin{\%} E_2 \ \mid\ E_1 \mathbin{\&} E_2 \ \mid\ E^{*} \end{array}$$

where $\mathcal{L}$ is the set of location names ranged over by $\ell, \ell_1, \ell_2, \ldots$, and where points correspond to the locations of suppliers and customers. We often omit $\mathtt{0}$. We describe a subset language of $\mathcal{E}$ as $\mathcal{S}$, when eliminating $E_1 \# E_2$, $E_1 \% E_2$, $E_1 \& E_2$, and E^{*} from $\mathcal{E}$. Let $S, S_1, S_2, \ldots$ be elements of $\mathcal{S}$. □

This framework assumes that each truck has its own route written in $\mathcal{S}$ and that its driver visits points along the route, i.e., intuitively, the meaning of the terms is as follows:

- $\mathtt{0}$ represents a terminated route.
- ℓ represents that a truck moves to a point called ℓ.
- $E_1 \mathbin{\mathtt{;}} E_2$ denotes the sequential composition of two routes E_1 and E_2. If the route of E_1 terminates, then the route of E_2 follows that of E_1.
- $E_1 + E_2$ represents the route of a truck according to either E_1 or E_2, where the selection is done by the truck.
- $E_1 \# E_2$ means that a truck itself can go through either E_1 or E_2.
- $E_1 \% E_2$ means that a truck can follow either E_1 before E_2 or E_2 before E_1 on its route.
- $E_1 \& E_2$ means that two routes, E_1 and E_2, may be executed asynchronously.
- E^{*} is a transitive closure of E and means that a truck may move along E an arbitrary number of times.

where in $E_1 + E_2$ the truck can select the E_1 (or E_2) route when the E_1 route is available. For example, if the E_1 route is available and the E_2 route is congested, the truck goes through the E_1 route. $E_1 \# E_2$ means that a truck can go through either E_1 or E_2. $E_1 \% E_2$, $E_1 \& E_2$, and E^{*} are used to specify possible routes. For example, $E_1 \# E_2$ permits the truck to go through one of the E_1 or E_2 routes.

To accurately express such routes, we need to define a specification language based on a process calculus approach such as CCS [6]. The semantics of the language are defined by the following labeled transition rules:

Definition 2 The language is a labeled transition system $\langle\, \mathcal{E},\, \mathcal{L} \cup \{\tau\}\, \{ \xrightarrow{\alpha} \subseteq \mathcal{E} \times \mathcal{E} \mid \alpha \in \mathcal{E} \cup \{\tau\}\, \}\, \rangle$ is defined as the induction rules below:

$$\frac{-}{\ell \xrightarrow{\ell} 0} \quad \frac{E_1 \xrightarrow{\ell} E_1'}{E_1 \mathbin{;} E_2 \xrightarrow{\ell} E_1' \mathbin{;} E_2} \quad \frac{E_1 \xrightarrow{\ell} E_1'}{E_1 + E_2 \xrightarrow{\ell} E_1'} \quad \frac{E_2 \xrightarrow{\ell} E_2'}{E_1 + E_2 \xrightarrow{\ell} E_2'}$$

$$\frac{E_1 \xrightarrow{\ell} E_1'}{E_1 \mathbin{\&} E_2 \xrightarrow{\ell} E_1' \mathbin{\&} E_2} \quad \frac{E_2 \xrightarrow{\ell} E_2'}{E_1 \mathbin{\&} E_2 \xrightarrow{\ell} E_1 \mathbin{\&} E_2'}$$

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \mathbin{;} E_2 \xrightarrow{\tau} E_1' \mathbin{;} E_2} \quad \frac{-}{E_1 \mathbin{\#} E_2 \xrightarrow{\tau} E_1} \quad \frac{-}{E_1 \mathbin{\#} E_2 \xrightarrow{\tau} E_2} \quad \frac{-}{E_1 \parr E_2 \xrightarrow{\tau} E_1 \mathbin{;} E_2}$$

$$\frac{-}{E_1 \parr E_2 \xrightarrow{\tau} E_2 \mathbin{;} E_1} \quad \frac{E_1 \xrightarrow{\tau} E_1'}{E_1 + E_2 \xrightarrow{\tau} E_1' + E_2} \quad \frac{E_2 \xrightarrow{\tau} E_2'}{E_1 + E_2 \xrightarrow{\tau} E_1 + E_2'}$$

$$\frac{E_1 \xrightarrow{\tau} E_1'}{E_1 \mathbin{\&} E_2 \xrightarrow{\tau} E_1' \mathbin{\&} E_2} \quad \frac{E_2 \xrightarrow{\tau} E_2'}{E_1 \mathbin{\&} E_2 \xrightarrow{\tau} E_1 \mathbin{\&} E_2'}$$

where $0 \mathbin{;} E$ is treated as being syntactically equal to E. $E^\star$ is recursively defined as $0 \mathbin{\#} (E \mathbin{;} E^\star)$. We often abbreviate $E_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} E_n$ to $E_0 (\xrightarrow{\tau})^n E_n$. $\square$

In Definition 2, the ℓ-transition defines the semantics of a trucks movement. For example $E \xrightarrow{\ell} E'$ means that the truck moves to a point named ℓ and then behaves as E'. Also, if there are two possible transitions $E \xrightarrow{\ell_1} E_1$ and $E \xrightarrow{\ell_2} E_2$ for a truck, the processing by the truck chooses one of the destinations, ℓ_1 or ℓ_2. In contrast, the τ-transition corresponds to a non-deterministic choice of a truck's routes .

Readers may think that the above operational semantics could be more compact. However, the aim is to design a system that can be easily implemented because the purpose of the framework is not to provide just a theoretical foundation for determining truck-route logistics, but a practical mechanism for selecting suitable trucks for milk-run operations. The language does not needs recursive or loop notations, because each truck does not continue to run for 24 hours everyday.

We show several basic examples of the language as shown in Figure 4.

– Route specification, $a \mathbin{;} b \mathbin{;} c \mathbin{;} d$, in $\mathcal{S}$ is interpreted as follows:

$$\begin{aligned} a \mathbin{;} b \mathbin{;} c \mathbin{;} d &\xrightarrow{a} b \mathbin{;} c \mathbin{;} d \\ &\xrightarrow{b} c \mathbin{;} d \\ &\xrightarrow{c} d \\ &\xrightarrow{d} \end{aligned}$$

The first diagram in Figure 4 illustrates the above derivation.

– Next, we show an example of a specification in $\mathcal{E}$. This is a route requirement.

$$\begin{aligned} a \mathbin{;} (b \mathbin{\#} c) \mathbin{;} d \mathbin{;} e &\xrightarrow{a} (b \mathbin{\#} c) \mathbin{;} d \mathbin{;} e \\ &\xrightarrow{\tau} b \mathbin{;} d \mathbin{;} e \quad \text{or} \quad c \mathbin{;} d \mathbin{;} e \end{aligned}$$

where $\#$ corresponds to a combination of two required routes so that trucks are required to follow both routes as shown in the third diagram in Figure 4. That is, a truck needs to call at point a and then at either b or c. Next, it calls at d and then e.

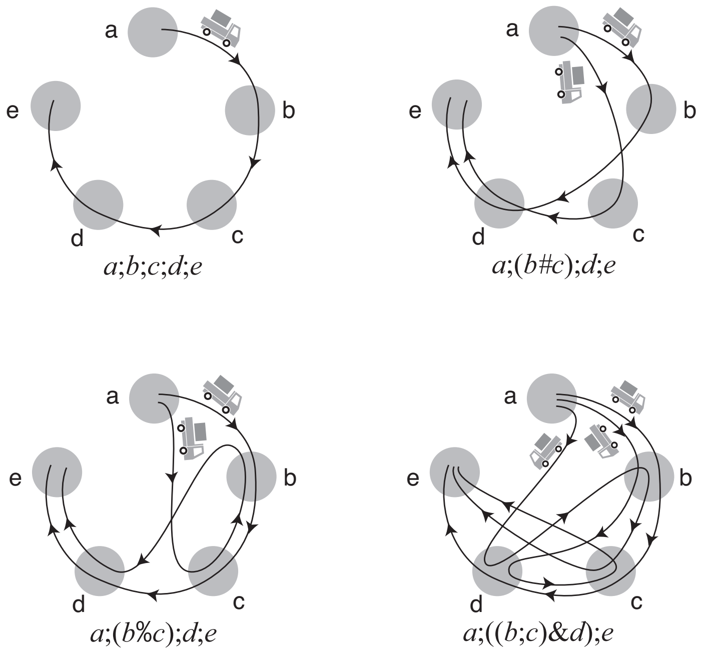

Fig. 4. Examples of specification.

- We show another route requirement specification, $a \mathbin{;} (b \mathbin{\%} c) \mathbin{;} d \mathbin{;} e$, in $\mathcal{E}$. It has two derivations as follows:

$$a \mathbin{;} (b \mathbin{\%} c) \mathbin{;} d \mathbin{;} e \xrightarrow{a} (b \mathbin{\%} c) \mathbin{;} d \mathbin{;} e$$
$$\xrightarrow{\tau} b \mathbin{;} c \mathbin{;} d \mathbin{;} e \quad \text{or} \quad c \mathbin{;} b \mathbin{;} d \mathbin{;} e$$

  where % means that trucks can take either one of the two routes before they take the other. The second diagram in Figure 4 shows possible routes that could satisfy this requirement specification.

- $a \mathbin{;} ((b \mathbin{;} c) \mathbin{\&} d) \mathbin{;} e$ in $\mathcal{E}$ is an example of & .

$$a \mathbin{;} ((b \mathbin{;} c) \mathbin{\&} d) \mathbin{;} e \xrightarrow{a} ((b \mathbin{;} c) \mathbin{\&} d) \mathbin{;} e$$
$$\xrightarrow{b} (c \mathbin{\&} d) \mathbin{;} e$$
$$\xrightarrow{c} d \mathbin{;} e$$
$$\xrightarrow{d} e$$

  where & corresponds to asynchronous reduction. Thus, this permits a truck to move to d while moving along $c \mathbin{;} b$. As shown in the fourth diagram in Figure 4, the following two derivations are possible in addition to the above derivation.

$$\begin{aligned} a \,;\, ((b \,;\, c) \,\&\, d) \,;\, e &\xrightarrow{a} ((b \,;\, c) \,\&\, d) \,;\, e \\ &\xrightarrow{b} (c \,\&\, d) \,;\, e \\ &\xrightarrow{d} c \,;\, e \\ &\xrightarrow{c} e \end{aligned}$$

or

$$\begin{aligned} a \,;\, ((b \,;\, c) \,\&\, d) \,;\, e &\xrightarrow{a} ((b \,;\, c) \,\&\, d) \,;\, e \\ &\xrightarrow{d} (b \,;\, c) \,;\, e \\ &\xrightarrow{b} c \,;\, e \\ &\xrightarrow{c} e \end{aligned}$$

– The first requirement presented in the previous section is described as specification $(a \,;\, (b \,\%\, c)) \,\&\, d^\star \,\&\, e^\star$. We show one of the possible derivations from the specification as follows:

$$\begin{aligned} (a \,;\, (b \,\%\, c)) \,\&\, d^\star \,\&\, e^\star &\xrightarrow{a} (b \,\%\, c)) \,\&\, d^\star \,\&\, e^\star \\ &\xrightarrow{b} c \,\&\, d^\star \,\&\, e^\star \end{aligned}$$

We can also have another derivation from the specification as follows:

$$\begin{aligned} (a \,;\, (b \,\%\, c)) \,\&\, d^\star \,\&\, e^\star &\xrightarrow{a} (b \,\%\, c)) \,\&\, d^\star \,\&\, e^\star \\ &\xrightarrow{c} b \,\&\, d^\star \,\&\, e^\star \end{aligned}$$

where $E \,\&\, d^\star$ means that the truck can visit d more than zero times while it moves along E.

$$\begin{aligned} (a \,;\, (b \,\%\, c)) \,\&\, d^\star \,\&\, e^\star &\overset{\text{def}}{=} (a \,;\, (b \,\%\, c)) \,\&\, (0 \,\#\, d \,;\, d^\star) \,\&\, e^\star \\ &\xrightarrow{tau} (a \,;\, (b \,\%\, c)) \,\&\, (d \,;\, d^\star) \,\&\, e^\star \\ &\xrightarrow{d} (a \,;\, (b \,\%\, c)) \,\&\, d^\star \,\&\, e^\star \end{aligned}$$

To describe routes in a compact notation, we define several macro notations for specifying the typical routes of trucks in a logistics operation. We describe a list of point names as $[\ell_1, \ell_2, \ldots, \ell_n]$, where $\ell_1, \ell_2, \ldots,\ \ell_n \in \mathcal{L}$. Let [] be an empty list, $car(X)$ be the top element of list X, i.e., ℓ_1, and $cdr(X)$ be the remaining list of X except for the top element, i.e., $[\ell_2, \ldots, \ell_n]$. Each point list is often written as \$(H) in terms of the language, where H is the name of a list, to avoid confusion between the name of a point and the name of the list. These macros do not extend the language because they are mapped into $\mathcal{E}$.

$$\begin{aligned} Cycle(\$(\mathrm{X})) &\overset{\text{def}}{=} car(\$(\mathrm{X})) \,;\, Cycle(cdr(\$(\mathrm{X}))) \\ Cycle([]) &\overset{\text{def}}{=} 0 \\ Star(\$(\mathrm{X})|\ell) &\overset{\text{def}}{=} (car(\$(\mathrm{X})) \,;\, \ell) \,;\, Star(cdr(\$(\mathrm{X}))|\ell) \\ Star([]|\ell) &\overset{\text{def}}{=} 0 \end{aligned}$$

Figure 5 illustrates $Cycle$ and $Star$ notations. Let ℓ be an element of $\mathcal{L}$ and X be a list of node names in $\mathcal{L}$. For example, $Cycle(\ \$(\texttt{DAIRY-FARMERS}))$ allows a truck to travel around the points specified in the `DAIRY- FARMERS` list consisting of the names of dairy farmers that produce milk; and f is a processing plant for dairy products. $Star(\$(\texttt{DAIRY-FARMERS})|f)$ corresponds to a star-shaped route, which allows a truck to go back and forth between the destinations specified in the `DAIRY-FARMERS` list and a given base point, e.g., a dairy factory, specified as f as the order of the list. To illustrate the transition defined in Definition 2, we show the transition of $Star$ $(\$(\texttt{DAIRY-FARMERS})|f)$ in $\$(\texttt{DAIRY-FARMERS}) = [a, b, c, d]$, where a, b, c, and d are the locations of dairy farmers as follows:

$$\begin{aligned}
Star(\$(\texttt{DAIRY-FARMERS})|f) \;\; &is \;\; Star([a,b,c,d]|f)\\
&\stackrel{\text{def}}{=} (a \,;\, f)\,;\, Star([b,c,d]|f)\\
&\xrightarrow{a} f \,;\, Star([b,c,d]|f)\\
&\xrightarrow{f} Star([b,c,d]|f)\\
&\stackrel{\text{def}}{=} (b \,;\, f)\,;\, Star([c,d]|f)\\
&\xrightarrow{b} f \,;\, Star([c,d]|f)\\
&\xrightarrow{f} Star([c,d]|f)
\end{aligned}$$

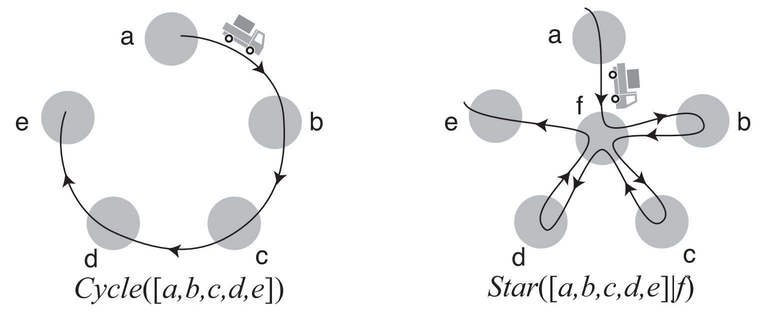

Fig. 5. Cycle and Star macros for routes

4 Order Relation for Route Selection

This section defines an order relation for selecting trucks according to their routes based on the concept of bisimulation [6]. The relation is suitable for selecting a truck for a milk-run operation with a route that satisfies the requirements of suppliers and customers.

Definition 3 A binary relation $\mathcal{R}^n$ ($\mathcal{R} \subseteq (\mathcal{E} \times \mathcal{S}) \times \mathcal{N}$) is an *n-route* prebisimulation, where $\mathcal{N}$ is the set of natural numbers, if whenever $(E, S) \in \mathcal{R}^n$ where $n \geq 0$, then, the following holds for all $\ell \in \mathcal{L}$ or τ.

i) if $E \xrightarrow{\ell} E'$ then there is an S' such that $S \xrightarrow{\ell} S'$ and $(E', S') \in \mathcal{R}^{n-1}$
ii) $E\,(\xrightarrow{\tau})^*\,E'$ and $(E', S) \in \mathcal{R}^n$
iii) if $S \xrightarrow{\ell} S'$ then there exist E', E'' such that $E\,(\xrightarrow{\tau})^* E' \xrightarrow{\ell} E''$ and $(E', S') \in \mathcal{R}^{n-1}$

where $E \sqsupseteq_n S$ if there exist some n-route prebisimulations such that $(E, S) \in \mathcal{R}^n$. We call the $\sqsupseteq_n$ *n-route* order. We often abbreviate $\sqsupseteq_n$ to $\sqsupseteq$. $\square$

The informal meaning of $E \sqsupseteq_n S$ is that S is included in one of the permissible routes specified in E and n corresponds to the number of movements of a truck that can satisfy E. We show several basic properties of the order relation below. Let us look at some basic examples.

- $(a \,\%\, b \,\%\, c)\ ;\ d \sqsupseteq_4\ c\ ;\ a\ ;\ b\ ;\ d$
  where the left-hand-side requires a truck to carry products to three points, a, b, and c in an indefinite order and then return to point d; the right-hand-side requires a truck to carry products to three points, c, a, and b, sequentially. When the left-hand-side is changed to $a\ ;\ b\ ;\ c\ ;\ d$, the relation is still preserved, but when the left-hand-side becomes $a\ ;\ d\ ;\ b\ ;\ d\ ;\ c\ ;\ d$ or $a\ ;\ b\ ;\ d$, the relation is not preserved.
- $((a\ ;\ b\ ;\ c)\ \&\ d^\star)\ ;\ d \sqsupseteq_6\ a\ ;\ d\ ;\ b\ ;\ d\ ;\ c\ ;\ d$
  where the left-hand-side allows a truck to drop in at point d an arbitrary number of times on route $a\ ;\ b\ ;\ c$ and then finish its movement at point d. The right-hand-side is a star-shaped route between three destinations, a, b, c, and point d satisfies the left-hand-side.
- $((a\ ;\ b)\ \&\ c^\star)\ \#\ ((b\ ;\ c)\ \&\ a^\star) \sqsupseteq_3\ a\ ;\ b\ ;\ c$
  where $((a\ ;\ b)\ \&\ c^\star)$ and $((b\ ;\ c)\ \&\ a^\star)$ on the left-hand-side are the required routes of two products, respectively. The first product is collected from point a and is then delivered to point b. The second product is collected from point b and is then delivered to point c. Both products permit trucks to visit c or a more then zero times. # on the left-hand-side means that trucks must satisfy both required routes. $a\ ;\ b\ ;\ c$ can satisfy the requirement specified on the left-hand-side. $b\ ;\ c\ ;\ a\ ;\ b$ still satisfies the left-hand-side, but the number of truck movements is greater than the $a\ ;\ b\ ;\ c$.

5 Implementation

This section describes a prototype implementation of our framework and a preliminary experiment using an RFID tag system. The experiment was constructed as a distributed logistics management system consisting of six supplier points in addition to a customer point with a route-selection server. Figure 6 shows the basic structure of the system. The server was responsible for receiving route requirements from suppliers and customers through a network and selecting suitable trucks with routes that satisfied these requirements.

5.1 Route selection algorithm

Here, let us explain the selection algorithm used for the current implementation, which we tried to make as faithful to Definition 3 as possible. The server maintains its own

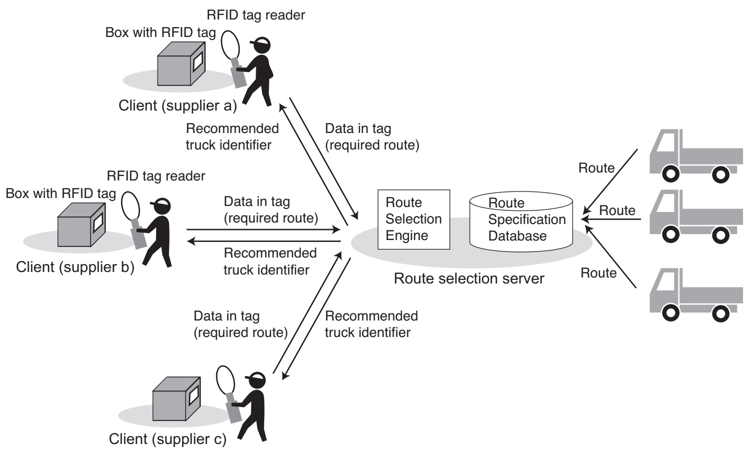
RFID tag reader
Box with RFID tag
Client (supplier a)
Recommended truck identifier
Data in tag (required route)
RFID tag reader
Box with RFID tag
Client (supplier b)
Data in tag (required route)
Recommended truck identifier
Route Selection Engine
Route Specification Database
Route selection server
Route
Route
Route
Data in tag (required route)
Recommended truck identifier
Client (supplier c)

Fig. 6. Basic structure of logistics management system

repository database containing the routes of trucks. To reduce the cost of the selection algorithm, the possible routes written in $\mathcal{E}$ are transformed into tree structures before they are stored in the database. These are called `transition trees` or `derivation trees` in the literature on process calculus [6]. Each tree is derived from a route in $\mathcal{E}$ according to Definition 2 and consists of arcs corresponding to ℓ-transitions or τ-transitions in the route. When a route selection server receives a required route from suppliers or customers, it extracts the required route written in $\mathcal{S}$ and then transforms the route into a transition tree. It next determines whether or not the trees derived from the routes stored in the database system can satisfy the tree derived from the required route by matching the two trees according to the definition of the order relation ($\sqsupseteq_n \subseteq \mathcal{E} \times \mathcal{S}$) as in the following.

(1) If each node in one of the two trees has arcs corresponding to ℓ-transitions, then the corresponding node in the other tree can have the same arcs, and the sub-nodes derived through the matching arcs of the two trees can still satisfy either (1) or (2).
(2) If each node in the tree derived from the required route has one or more arcs corresponding to τ-transitions, then at least one of the nodes derived through the arcs and the corresponding node in the tree derived from the truck's route can still satisfy (1) or (2).
(3) If neither (1) nor (2) is satisfied, the route selection server backtracks from the current nodes in the two trees and tries to apply (1) or (2) to their two backtracked nodes.

Figure 7 illustrates the matching of two transition trees in the above algorithm. If one or more truck routes in the database satisfy the required route, it selects the truck with the least number of truck movement between points, which is n of $\sqsubseteq_n$ in Definition 3. Although the cost of selecting a route is dependent on the number of trucks and the length of their routes, the system can handle each of the routes presented in this paper within a few milliseconds.

Non-deterministic operators, e.g., # and % , tend to cause the exposition of a number of sub-trees in transition trees. Nevertheless, our algorithm can easily restrain the number of sub-trees resulting from non-deterministic operators because the expansion rules of expressions, i.e., the operational semantics of the language, distinguish between derivations resulting from deterministic operators and those resulting from non-deterministic operators. Readers may wonder why E^* operator creates an infinite number of sub-trees, but the current implementation interprets the operator in a lazy evaluation manner.

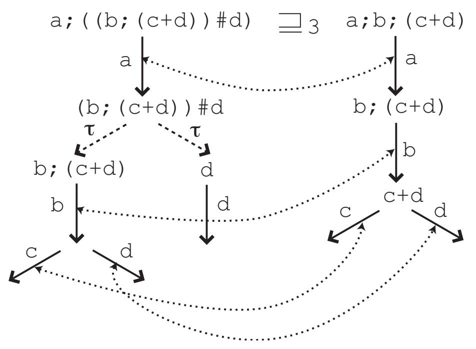

Fig. 7. Matching transition trees in route-order relation algorithm.

5.2 Route specification in RFID tags

The current implementation assumes that the routes required for products or pallets are stored in RFID tags attached to the products or pallets because they may have their own delivery requirements. The current implementation supported three commercial passive RFID tags systems: OMRON V700 RFID system (125 kHz), Phillips i-Code system (13.56 MHz), and Texas Instruments Tag-It systems (13.56 MHz). The first system provides each tag with 240 bytes, the second with 112 bytes, and the third with 32 bytes. We were able to maintain each of the example routes presented in these papers in the first and second tag systems, where the length of the identifier for each point was 4 bytes. Tags in the third system may not be able to store route specifications internally, but can maintain references to route specifications stored in a database server.

5.3 Early Experience

Our prototype implementation assumes that each client-side system at a supplier or customer point has more than one RFID tag reader. The reader periodically or explicitly tries to detect the presence of tags within its coverage area. Figure 8 shows a 13.56-MHz-based RFID tag reader embedded with a WiFi network interface scanning a tag attached to a pallet in a warehouse The tag specifies the route required to deliver the

Fig. 8. RFID tag reader scans route specification from tag

pallet. When it reads the data from a tag, it sends the route stored in the tag to the route-selection server via WiFi and waits for a response from the server. When it receives a truck identifier from the server, it displays the identifier on its screen.

The current implementation of the algorithm was not optimized for performance. Nevertheless, we describe the basic performance of the implementation. The cost of reading the route specification in a tag depends on the length of the specification, e.g. the cost of reading a specification with a length of less than 40 bytes is within 0.2 sec. When the routes of five trucks were registered in the server running on a computer (Intel Core 2 Duo 2 GHz and Windows XP), the cost of selecting a truck after the reader had detected a tag, including the cost of communication between the server and client via a TCP/IP session, was less than 1.2 sec. Client-side systems for suppliers and customers can be operated using only RFID readers, which connect to a server through either wired or wireless networks. This means they do not need any special equipment to use the logistics management system. This is important because in milk-run logistics, most suppliers are small to medium enterprises that do not want to have to invest in additional equipment for milk-run logistics.

6 Related Work

There have been many attempts to use process calculi, e.g., as formal methods for various business enterprise processes. Several researchers have used process calculi, e.g., π-calculus, as business-process modeling languages, such as BPEL, [13,4,8,12]. π-calculus has been used as a formal composition language for software composition and Web service composition, e.g., Orc [7] and SCC [1]. Process calculi are theoretically sound and support bisimulation analysis and model checking. They are also gaining increasing acceptance as a support tool in industry. However, there have been no process-calculus-based formal methods for logistics, in particular for improving the transport efficiency of trucks.

Several papers have explored formal models for specifying and reasoning about mobile agents, e.g., Mobile UNITY [5], Ambient calculus [2], and Join-calculus [3]. Ambient calculus [2] allows mobile agents (called ambients in the calculus) to contain other agents and to move with all inner ambients. The calculus must always model the mobility of agents as navigation along a hierarchy of agents, whereas the itineraries of real mobile agents may be more complicated. Join-calculus [3] also introduces the notion of named locations that form a tree. The mobility of an agent is modeled as a transformation of sub-trees from one part of the tree to another. The author presented a formal method for using mobile agents in network management systems [11]. However, this method was aimed only at mobile agents and assumed the notion of two-layer mobile agents.

7 Future Work

This section discusses further issues that need to be resolved. This paper assumes that trucks are independent, but coordination of multiple trucks is often necessary to ensure efficient transportation. In future research, we are interested in developing a mechanism for dividing single routes into multiple sub-routes and assigning these subtasks to one or more trucks. The order relation proposed in this paper can select truck routes according to the number of movements between points as well as the order in which the trucks visit each point. The amount of CO_2 emissions resulting from transport depends on the distance covered by trucks. We need to introduce the notion of distance into the framework. Although the milk-run approach is useful for non-just-in-time logistics, we are interesting in extending the framework by incorporating the ability to reason about time constraints. The language itself is general. We developed a methodology for testing software for mobile terminals that can be carried between networks and reconnected to current networks [9]. We plan to use the language as a control language for testing software in the methodology. As mentioned previously, the goal of the proposed framework is to establish both a theoretical and practical foundation for earth-friendly logistics. In fact, we have already started some large-scale experiments with logistics and warehouse companies in collaboration with the Ministry of Land, Infrastructure and Transport in Japan, to demonstrate the effectiveness of the proposed framework. The author is a member of the ISO/IEC standardization committee (SC31) for RFID tags and barcodes and supported several logistic services, including milk-run based logistic operations. We believe the the language can provide a foundation for tags and barcodes for earth-friendly logistic systems.

8 Conclusion

We presented a formal method for improving transport efficiency, using the example of milk-run logistics, to reduce the environmental impacts of transport operations. The method was formulated based on a process calculus-based language and an order relation over two terms corresponding to truck routes and the required routes in the language. The language can specify truck routes for milk-run operations and the required routes for shipping. The relation can be used to accurately determine whether a truck route satisfies the requirements of customers and suppliers. A prototype implementation system based on the framework was constructed using Java language and RFID tag systems and applied to our experimental distributed logistics management system.

References

1. M. Boreale, R. Bruni, L. Caires, R. De Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, D. Sangiorgi, V. T. Vasconcelos, and G. Zavattaro, SCC: a Service Centered Calculus, Proceedings of Web Services and Formal Methods, LNCS Vol.4184, pp.38-57, Springer, September 2006.
2. L. Cardelli and A. D. Gordon, Mobile Ambients, Proceedings of Foundations of Software Science and Computational Structures, LNCS, Vol. 1378, pp. 140-155, 1998.
3. C. Fournet, G. Gonthier, J. Levy, L. Marnaget, and D. Remy, A Calculus of Mobile Agents, Proceedings of CONCUR'96, LNCS, Vol. 1119, pp.406-421, Springer, 1996.
4. M. Mazzara and R. Lucchi. A pi-calculus based semantics for WS-BPEL. Journal of Logic and Algebraic Programming, vol.70, no.1, pp.96-118, 2006.
5. P.J. McCann, and G.-C. Roman, Compositional Programming Abstractions for Mobile Computing, IEEE Transaction on Software Engineering, Vol. 24, No.2, 1998.
6. R. Milner, Communication and Concurrency, Prentice Hall, 1989.
7. J. Misra and W. R. Cook, Computation orchestration: A basis for wide-area computing, Journal of Software and Systems Modeling, 2006. (A preliminary version of this paper appeared in the Lecture Notes for NATO summer school, August 2004)
8. F. Puhlmann and M. Weske, Using the Pi-Calculus for Formalizing Workow Patterns, Proceedings of the International Conference on Business Process Management, pp.153-168, 2005
9. I. Satoh, A Testing Framework for Mobile Computing Software, IEEE Transactions on Software Engineering, vol. 29, no. 12, pp.1112-1121, December 2003.
10. I. Satoh, A Location Model for Pervasive Computing Environments, Proceedings of IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05), pp,215-224, IEEE Computer Society, March 2005.
11. I. Satoh, Building and Selecting Mobile Agents for Network Management, Journal of Network and Systems Management, vol.14, no.1, pp.147-169, Springer, 2006.
12. H. Smith, Business Process Management-The Third Wave: Business Process Modeling Language (BPML) and Its Pi-Calculus Foundations, Information and Software Technology 45, No. 15, 1065-1069, 2003.
13. K. Xu, Y. Liu, J. Zhu, and C. Wu, Pi-Calculus Based Bi-transformation of State-Driven Model and Flow-Driven Model, International Journal of Business Process Integration and Management, 2006.