PAPER Special Section on Parallel and Distributed Computing and Networking

Automated Route Planning for Milk-Run Transport Logistics with the NuSMV Model Checker*

Takashi KITAMURA[†a)], *Member and* Keishi OKAMOTO[††], *Nonmember*

SUMMARY In this paper, we propose and implement an automated route planning framework for milk-run transport logistics by applying model checking techniques. First, we develop a formal specification framework for milk-run transport logistics. The framework adopts LTL (Linear Temporal Logic), a language based on temporal logics, as a specification language for users to be able to flexibly and formally specify complex delivery requirements for trucks. Then by applying the bounded semantics of LTL, the framework then defines the notion of "optimal truck routes", which mean truck routes on a given route map that satisfy given delivery requirements (specified by LTL) with the minimum cost. We implement the framework as an automated route planner using the NuSMV model checker, a state-of-the-art bounded model checker. The automated route planner, given route map and delivery requirements, automatically finds optimal trucks routes on the route map satisfying the given delivery requirements. The feasibility of the implementation design is investigated by analysing its computational complexity and by showing experimental results.

key words: *automated route planning, transport logistics, model checking*

1. Introduction

Milk-run transport logistics, which refers to the means of transportation where a single truck cycles around multiple suppliers to collect or deliver freights, is one of the most efficient and popular approaches to improve logistic operations. Recently, a variety of industries, e.g., food, automobile manufacturing, military, as well as the dairy industry, have adopted the milk-run approach to make their logistics operations more efficient. However, compared with the existing (non-efficient) logistics, logistics operations with milk-run logistics often tend to be complex, and it is one of the main barriers preventing wider prevalence of the approach in industry.

To address the complexity aspect, Satoh in [1], [2] proposed a novel framework for milk-run transport logistics. Focusing on the complexity of truck routes which is a characteristic of the logistics operations, the framework introduces a formal specification language that can flexibly specify complex truck routes, by specifying complex order of locations which trucks should visit. Further it provides a mechanism for selecting appropriate trucks according to the specified truck routes. The framework is realized by applying theory and techniques of process algebra; the specification language is designed based on CCS (Calculus of Communicating System) [3], and the mechanism for selecting appropriate trucks is realized by applying the notion of bisimulation checking [3].

Inspired by [1], [2] we develop a framework for *automated route planning* for such milk-run logistics. The framework, given delivery requirements and a route map, deals with *optimal truck routes* on the route map that satisfy the given delivery requirements. We apply a model checking approach to realize this framework. The framework adopts LTL (Linear Temporal Logic) [4], [5] as a specification language for users to be able to flexibly specifying the delivery requirements in milk-run logistics, which are often complex w.r.t. the locations order of truck routing. Then by applying the bounded semantics of LTL, the framework defines the notion of "optimal truck routes", which mean truck routes on a given route map that satisfy given delivery requirements (specified by LTL) with the minimum cost.

We will also develop a prototype of the framework as an automated route planner. We implement the prototype using the NuSMV model checker, which is one of the state-of-the-art LTL bounded model checkers. In doing so, we will explain several gaps lying between the framework and the model checker, and techniques to bridge them. we evaluate the feasibility of the implementation design by analysing its computational complexity and showing experimental results using NuSMV. In our previous paper [6], we tackled on developing a theoretical foundation of the framework, and this paper focuses on implementation of the framework based on the foundation and evaluating the framework and its implementation design.

We end this section with the outline of the paper: the next section explains the background of the framework we develop, Sect. 2 briefly reviews LTL, Sect. 3 explains the details of our framework, Sect. 4 explains our prototype implementation of the framework for automated route planning using NuSMV, and the conclusion and discussion of future work follow in the last section.

2. Background

2.1 Example Scenario

Consider a route map as a weighted and undirected graph;

Manuscript received January 12, 2013.
Manuscript revised June 20, 2013.

[†]The author is with National Institute of Advanced Industrial Science and Technology (AIST), Amagasaki-shi, 661–0974 Japan.
[††]The author is with Sendai National College of Technology, Sendai-shi, 989–3128 Japan.

[*]The previous version of this paper was presented at 4th International Workshop on Parallel and Distributed Algorithms and Applications (PDAA2012).

a) E-mail: takashi.kitamura@gmail.com

DOI: 10.1587/transinf.E96.D.2555

Table 1 Different truck routes (1) (2) and (3) to satisfy/violate the delivery requirements.

(1)	$a^0 \to^4 b^4 \to^2 e^6 \to^2 b^8 \to^3 d^{11} \to^2 h^{13} \to^1 i^{14} \to^2 o^{16} \to^3 n^{19} \to^2 q^{21} \to^2 t^{23} \to^3 p^{26} \to^2 l^{28} \to^1 k^{29} \to^1 l^{30} \to^1 m^{31} \to^1 j^{32} \to^1 g^{33} \to^2 d^{35} \to^2 a^{37} \to^2 c^{39}$
(2)	$a^0 \to^4 b^4 \to^3 d^7 \to^2 h^9 \to^2 o^{11} \to^3 n^{14} \to^3 m^{17} \to^1 n^{18} \to^1 q^{19} \to^2 t^{21} \to^2 p^{23} \to^3 l^{25} \to^2 k^{27} \to^1 e^{28} \to^5 b^{33} \to^4 a^{37} \to^4 c^{39} \to^3 i^{42}$
(3)	$a_0 \to^2 d_2 \to^2 h_4 \to^3 o_7 \to^3 h_{10} \to^2 d_{12} \to^2 b_{14} \to^2 e_{16} \to^3 f_{19} \to^1 m_{20} \to^1 l_{21} \to^1 k_{22} \to^1 l_{23} \to^2 p_{25} \to^2 t_{27} \to^3 q_{30} \to^2 n_{32} \to^2 g_{34} \to^2 d_{36} \to^2 c_{38}$

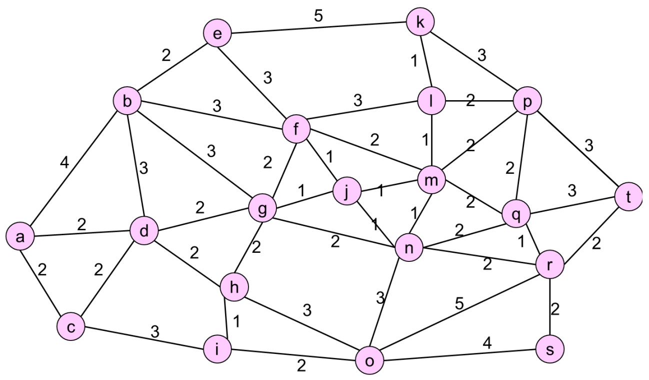


Fig. 1 An example of route map.

i.e., the nodes, the edges and the weights of the weighted graphs express the locations, the routes, and the costs of a truck moving on the route segment in the route map. Figure 1 shows an example of such a route map.

"Delivery requirements" mean conditions for delivery regarding the truck routes; i.e., trucks have to satisfy these conditions on their routes through route map. A main characteristic of delivery requirements in milk-run logistics typically appears in the complex order of locations which trucks need to visit. This is a consequence of an important feature of milk-run logistics: trucks are shared by multiple users (i.e., suppliers and customers). Trucks collect freights at one or more source points and deliver them to one or more destination points on their way, visiting the source points before the destination points. Typical delivery requirements in milk-run logistics are exemplified as follows:

i) A truck visits location b and then k, and after that visits location a or i and then s.

ii) A truck visits location e and then i, but between them it should not go through f and o.

iii) A truck visits location o and then t and p and then c, while visiting from o to t it should not go through r.

Given such a route map and delivery requirements, we discuss the truck routes on the route map that satisfy and violate the given requirements. Further, we can discuss the truck routes with the best efficiency, i.e., the truck routes satisfying the requirements with the smallest cost, which we call "*optimal truck routes*".

Assume that a truck has sufficient carrying capacity, and it starts at location a. Table 1 shows three trucks routes on the route map that satisfy and violate the delivery requirements on different routes. Each truck route is represented as a sequence of locations, where locations are connected by $\to$. The sequence of the locations is the order in which the trucks visit the locations. The (natural) number attached to each arrow expresses the cost for a truck moving from previous to next location, and the number on each location expresses the accumulated cost to reach the location on the truck route, given the route map and sequence.

Observe that truck route (1) and (2) satisfy the above delivery requirements (i)–(iii), and truck route (3) does not. Also, we consider that truck route (1) is more efficient than (2); (1) requires cost 39 on its route to satisfy the given requirements while (2) requires 42. In fact, (1) is an *optimal truck route*, which is a truck route on the route map that satisfies the delivery requirements with the minimum cost. Note that the cost of a truck route is the sum of the weights on edges in its route, instead of the number of locations which it goes through. Accordingly, though truck route (2) visits less locations on its route than (1), (1) is considered to be more efficient.

2.2 Requirements for the Framework

With this example scenario, we clarify the main requirements of the framework for automated route planning for milk-run logistics. We distinguish *general* requirements for milk-run logistics and *specific* ones for our framework of automated route planning. We share *general* requirements for milk-run logistics in common from [1], [2] as follows:

- *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.*
- *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 additions to the *general* requirements for milk-run logistics, we assume the following *specific* ones for our framework:

- The framework, given a route map and delivery requirements, provides a mechanism to find optimal truck routes, which are truck routes satisfying the given delivery requirements with the minimum cost. Also, the framework provides a mechanism, given a cost, to find a truck route with that cost, and additionally to check if there exists a truck route which satisfies the given requirements with that cost.
- To keep generality, route maps are regarded as weighted undirected graphs, following the precedent of general routing problems such as [7]. Weights are imposed on the edges, expressing a notion of cost, interpreted as, e.g., time, money, and emission of CO_2, for trucks' moving on the edges.
- Delivery requirements in milk-run logistics are complex w.r.t. the order of locations which trucks visit. Also, the requirements vary temporally, since users have different requirements each year, each month, and each week, depending on their use. To address these aspects, the framework should take a language-based approach for specifying delivery requirements; i.e., it should provide a language with which users can flexibly specify delivery requirements.

2.3 Basic Approach

Our aim is to develop a framework for milk-run logistics to satisfy the above requirements. We realize the framework applying model-checking techniques. Our basic approach is to specify delivery requirements with LTL and route maps with Kripke models, and apply model checking algorithms to find truck routes that best satisfy the requirements. That is, we interpret automated route planning as the following model checking problem:

route map $\models$ *delivery requirements*.

As the framework takes a language-based approach to specify delivery requirements, we adopt LTL for the purpose. LTL is a logical language that can specify complex temporal properties by means of ordered events to happen. And it makes an appropriate basis for specifying complex and various delivery requirements in milk-run logistics. For example, delivery requirement (i) (ii) and (iii) can be specified with LTL as;

i) $\mathbf{F}(b \wedge \mathbf{F}(k \wedge \mathbf{F}(a \vee (i \wedge \mathbf{F}s))$
ii) $\mathbf{F}(e \wedge \neg(f \vee o)\mathbf{U}i)$
iii) $\mathbf{F}(o \wedge (\neg r\mathbf{U}t) \wedge \mathbf{F}(p \wedge \mathbf{F}c))$

By applying the semantics of LTL, the framework preserves formal accounts of the satisfaction or violation of truck routes given complex delivery requirements. To realize the framework for automated route planning, we apply *bounded semantics* and *bounded model checking (BMC)* proposed by [4], [5]. Due to this approach, the framework can impose a cost bound into the analysis of a truck route satisfying or violating delivery requirements. Consequently, the framework can formally define "optimal truck routes", which are truck routes that satisfy the delivery requirements with the minimum cost. Further, we implement the framework using NuSMV, an off-the-shelf BMC model checker.

We adopt LTL in this paper instead of CTL, which is the other of the two major basic temporal logics, though we consider that a similar discussion with LTL in this paper can be porting to that with CTL basically. This is because we consider LTL is relatively more suitable than CTL for our purpose. Its reason is the following two fold.

The first reason is due to that the semantics design of LTL is more naturally fit for specifying truck routes in our setting. Roughly speaking, the LTL semantics, which is a linear temporal logic, interprets a temporal formula on each fixed single path, which is a sequence of states on a Kripke model (or generally, a graph). On the other hand, the CTL semantics, which is a branching temporal logic, interprets a temporal formula on a tree, which can be regarded as a set of all possible paths. Hence, CTL is equipped with the universal existential path quantifier (the $\forall$ and $\exists$ operator), which respectively expresses all possible branches and only one branch, when facing branches on the tree. Interpretation of satisfiability of a truck route to delivery requirements, as exemplified above, more naturally fit the LTL semantics. Our setting assumes that a truck route is a single path in a route map like LTL, and do not assume to specify if a set of truck routes can satisfy all branches in a route map, specified by universal path quantifiers as in CTL.

The second reason, which is more practical, is due to that the tool we use in this work is more advanced for LTL model checking than CTL model checking. This work aims not only to develop a formal framework for milk-run logistics but also to evaluate feasibility by implementing the framework. For the implementation, we use the NuSMV model checker instead of implementing a model checker from scratch. NuSMV is one of the state-of-the-art model checking tools, and its semantic design is based on the Kripke model which the framework we develop is based on and hence is appropriate. And the model checking tool is deviced with more advanced features for LTL model checking, with more varieties of functions. In implementing the framework, we use these advanced features of LTL model checking in order to bridge gaps lying when applying a software verification tool to automated route planner. We will explain the details in Sect. 5.

3. Bounded LTL Model Checking

Bounded Semantics of LTL [4], [5], [8], [9]

Let AP be a set of atomic propositions., ranged over by $p, q, \cdots$. Then LTL formulas over AP are defined recursively as follows: atomic propositions are LTL formulas; and if ϕ_1 and ϕ_2 are LTL formulas so are $X\phi$ (neXt), $\phi_1 U \phi_2$ (ϕ_1 Until ϕ_2), $\phi_1 \vee \phi_2$ and $\neg\phi_1$. A Kripke model M over AP is a quadruple $M = (S, I, T, L)$ where S is a finite set of states,

Table 2 Bounded LTL semantics without a loop.

$\pi \models_k^i p$	iff $p \in L(\pi(i))$
$\pi \models_k^i \neg p$	iff $p \notin L(\pi(i))$
$\pi \models_k^i \phi_1 \wedge \phi_2$	iff $\pi \models_k^i \phi_1 \wedge \pi \models_k^i \phi_2$
$\pi \models_k^i \phi_1 \vee \phi_2$	iff $\pi \models_k^i \phi_1 \vee \pi \models_k^i \phi_2$
$\pi \models_k^i \mathbf{X}\phi$	iff $i < k \wedge \pi_1 \models_k^{i+1} \phi$
$\pi \models_k^i \mathbf{F}\phi$	iff $\exists j, i \leq j \leq k.\pi \models_k^j \phi$
$\pi \models_k^i \mathbf{G}\phi$	*is always false.*
$\pi \models_k^i \phi_1 \mathbf{U} \phi_2$	iff $\exists j.i \leq j \leq k.\pi \models_k^j \phi_2 \wedge$ $\forall h, i \leq h < j.\pi_i \models_k^h \phi_1$
$\pi \models_k^i \phi_1 \mathbf{R} \phi_2$	iff $\exists j.i \leq j \leq k.\pi \models_k^j \phi_1 \wedge$ $\forall h, i \leq h < j.\pi_i \models_k^h \phi_2$

Table 3 Bounded LTL semantics for a route map.

$\dot{1}:$	$\pi \models_k^i l$	iff $\pi(i) = l$
$\dot{2}:$	$\pi \models_k^i \neg l$	iff $\pi(i) \neq l$
$\dot{3}:$	$\pi \models_k^i \phi_1 \wedge \phi_2$	iff $\pi \models_k^i \phi_1 \wedge \pi \models_k^i \phi_2$
$\dot{4}:$	$\pi \models_k^i \phi_1 \vee \phi_2$	iff $\pi \models_k^i \phi_1 \vee \pi \models_k^i \phi_2$
$\dot{5}:$	$\pi \models_k^i \mathbf{X}\phi$	iff $i < k \wedge \pi_1 \models_k^{i+1} \phi$
$\dot{6}:$	$\pi \models_k^i \mathbf{F}\phi$	iff $\exists j, i \leq j \leq k.\pi \models_k^j \phi$
$\dot{7}:$	$\pi \models_k^i \mathbf{G}\phi$	iff $\forall j, i \leq j \leq k.\pi \models_k^j \phi$
$\dot{8}:$	$\pi \models_k^i \phi_1 \mathbf{U} \phi_2$	iff $\exists j.i \leq j \leq k.\pi \models_k^j \phi_2 \wedge$ $\forall h, i \leq h < j.\pi_i \models_k^h \phi_1$
$\dot{9}:$	$\pi \models_k^i \phi_1 \mathbf{R} \phi_2$	iff $\exists j.i \leq j \leq k.\ pi \models_k^j \phi_1 \wedge$ $\forall h, i \leq h < j.\pi_i \models_k^h \phi_2$

$I \subseteq S$ is a finite set of initial states, $T \subseteq S \times S$ is the transition relation and $L : S \rightarrow 2^{AP}$ is the labeling function.

The LTL semantics is defined by way of paths of a Kripke model M. Besides, in bounded semantics, paths with loop-back and without loop-back are considered separately. Here, we only review the case for paths without loop-back since it is of the only situation necessary in our framework. A path π in M is a sequence $\pi = (s_0, s_1, \ldots)$ of states, given in an order that respects the transition relation of M. For $i < |\pi|$, $\pi(i)$ denotes the i-th state s_i in the sequence, and $\pi_i = (s_i, s_{i+1}, \cdots)$ denotes the suffix of π starting with state s_i. For $k \geq 0$, let π be a path without loop-back, and ϕ be a LTL formula. Then π satisfies ϕ with bound k (written as $\pi \models_k \phi$) iff $\pi \models_k^0 \phi$ where $\pi \models_k^i \phi$ is defined in Table 2.

It is defined that a Kripke model M satisfies a LTL formula ϕ, written $M \models \phi$, if $\pi \models \phi$ for all paths π of M; i.e., $M \models \phi$ *if* $\forall \pi \in M.\pi \models \phi$.

Bounded Model Checking Using SAT

One advantage of BMC is that we can discuss the semantics taking the length of paths into account. As an application, we can discuss paths that satisfy/violate an LTL formula with the shortest length. BMC problems are typically solved by reducing SAT problems, therefore the complexity of BMC of this approach is determined using the number of propositional variables to appear in formula by SAT encoding [8], [9]. The translation results in $O(k \cdot |\log(S)| + (k+1)^2 \cdot |\phi|)$ variables, where S is the number of states in model M, k is the bound, and $|\phi|$ is the length of ϕ.

4. An Automated Route Planning Framework for Milk-Run Logistics

In this section, we explain our framework for an automated route planning framework for milk-run logistics.

4.1 Semantics of Delivery Requirements w.r.t. Route Maps

First, we formally define what it means for a truck route in a route map to satisfy given delivery requirements, by providing a formal semantics of delivery requirements (specified by LTL) w.r.t. route maps. Though the semantics is designed based on bounded LTL semantics w.r.t. Kripke models, its significance is emphasised to accurately explain the framework. Also, a prototype implementation for the framework explained in later sections is developed based on the formal foundation. We start by defining route maps:

Definition 1 (route map): A *route map* M is a pair $M = (L, R)$, where L is a finite set of locations and $R (\subseteq (L \times L) \times \mathcal{N}^+)$ is a finite set of *routes* where $\mathcal{N}^+$ is the set of positive integers, satisfying the conditions: (1) If $((a, b), n) \in R$ and $((a, b), n') \in R$ then $n = n'$, and (2) If $((a, b), n) \in R$ then $((b, a), n) \in R$.

Like LTL semantics w.r.t. Kripke models, LTL semantics w.r.t. route maps are given in two steps; i.e., first semantics w.r.t. a path of a route map is given, then based on that, semantics w.r.t. route maps is given. To this end, the notion of truck routes, which we also call *paths*, on route maps is defined.

Definition 2 (Paths of a route map): A path π of a route map $\mathcal{M}$ is a sequence of pairs of a location and a natural number; i.e., $\pi = ((a_1, k_1), (a_2, k_2), (a_3, k_3), \cdots)$, where the natural number expresses the accumulated cost to reach the associated location, given in an order that aligns with the route map M. $\pi(k)$ denotes the location name paired with cost k in the path. Note that $\pi(k)$ may not be defined for all costs k. Thus, we consider π as a partial function, which, given a cost, returns a location name. $\pi_l(i)$ and $\pi_k(i)$ denote the location and the accumulated cost at the i-th pair in path π, respectively. For a brief notation, we may also write a path as "$\pi_l(1)^{\pi_k(1)} \rightarrow^{n_1} \pi_l(2)^{\pi_k(2)} \rightarrow^{n_2} \pi_l(3)^{\pi_k(3)} \ldots$", where $n_i = \pi_k(i+1) - \pi_k(i)$.

The path representations in Table 1 follow this definition using the brief notion.

Next the semantics of delivery requirements w.r.t. paths of a route map is defined. It is designed as a bounded semantics: i.e., it introduces a natural number to express a cost bound, and defines a path that satisfies delivery requirements taking the cost bound into account.

Definition 3: (Bounded semantics w.r.t. paths of route maps): Assume M is a route map, π a path of M, and ϕ

an LTL formula. For a cost bound k such that $k \geq 0$, we define that π satisfies ϕ with bound k, denoted by $\pi \models_k \phi$, if $\pi \models_k^0 \phi$, where $\pi \models_k^i \phi$ is defined as Table 3.

The rules given in Table 3 are same except for the **G** operator. For the **G** operator, due to this leap from the original bounded semantics of LTL w.r.t. Kripke models, we acquire duality between **G** and **F**, i.e., $\neg\mathbf{F}\phi \equiv \mathbf{G}\neg\phi$, in the context of the bounded semantics. We will use this property of the semantics in our implementation explained in Sect. 4.

Thus, the semantics of LTL w.r.t. route map M is defined. Since our interest is to find some truck-routes to satisfy the given delivery requirements, we define that if some path of the route map satisfies the delivery requirements, then the satisfaction relation holds.

Definition 4 (Semantics w.r.t. a route map): Let M be a route map and ϕ a LTL formula. We say a routemap M satisfy a LTL formula ϕ, written $M \models_k^\exists \phi$, iff $\pi \models_k^\exists \phi$ for *some* paths π of M,

The LTL semantics, in general, can be classified from the existential and universal viewpoint; i.e., determining whether an LTL formula ϕ is existentially/universally valid in a given model is called existential/universal model checking (EMC/UMC), respectively. For clarity, we use the notational convention of quantifier symbols $M \models^\exists \phi$ and $M \models^\forall \phi$ to denote them.

4.2 Examples of Specification Framework

This sub-section demonstrates how the semantics given in the previous sub-section interprets delivery requirements specified as LTL formula. First, we demonstrate the process of the semantics interprets a delivery requirement (ϕ) w.r.t. a path of route map (π); i.e., $\pi \models_k^\exists \phi$. The semantics, in other words, evaluates a given satisfaction relation of a path to delivery requirements to **True** or to **False**, which respectively means the path satisfies the delivery requirement or fails to satisfy it. So we show the process to evaluate the satisfaction relation to **True** or to **False**, which we may also call "satisfiability checking".

To demonstrate it, we only show that truck route $\pi_{(1)}$, which is truck route (1) in Table 1, satisfies delivery requirements (iii) with the cost of 39 for lack of space (i.e., $\pi_{(1)} \models_{39}$ (iii)), instead of delivery requirements (i), (ii), and (iii) (i.e., $\pi_{(1)} \models_{39}$ (i) $\wedge$ (ii) $\wedge$ (iii)). But the satisfiability checking of $\pi_{(1)} \models_{39}$ (iii) can be easily extended to that of $\pi_{(1)} \models_{39}$ (i) $\wedge$ (ii) $\wedge$ (iii). The process of satisfiability checking for $\pi_{(1)} \models_{39}$ (iii) is shown in Table 4, and is briefly explained below. (The satisfiability checking can be understood in a similar way as the standard discussion of that for model checking, found such as in [14], [16].)

The process proceeds from the top to the bottom, where the proposition, represented as a satisfaction relation, on the top expresses the target that we want to check for its satisfiability. The process goes down step by step from the top, where either one of the rules in Table 3 or a logical equivalence rule is applied to each step for going down. In the process in Table 4, when a rule in Table 3 is applied for a step, the name of the rule together with the symbol is noted to clarify the rule used in the step. Also, the symbol for logical equivalence ($\Leftrightarrow$) is used when such a rule is applied for a step. The process goes step by step by iteratively applying these rules, and such a process will lead a legitimate satisfaction relation to either of **True** or **False**.

Table 4 The process for satisfiability checking $\pi_{(1)} \models_{39} \mathbf{F}(o \wedge (\neg r\mathbf{U}t) \wedge \mathbf{F}(p \wedge \mathbf{F}c))$.

	$\pi_{(1)} \models_{39}^0 \mathbf{F}(o \wedge (\neg r\mathbf{U}t) \wedge \mathbf{F}(p \wedge \mathbf{F}c))$	
iff	$\pi_{(1)} \models_{39}^{16} o \wedge \neg r\mathbf{U}t \wedge \mathbf{F}(p \wedge \mathbf{F}c)$	(*by* $\dot{6}$)
iff	$\pi_{(1)} \models_{39}^{16} o \wedge \pi_{(1)} \models_{39}^{16} \neg r\mathbf{U}t \wedge \mathbf{F}(p \wedge \mathbf{F}c)$	(*by* $\dot{3}$)
iff	$\mathbf{True} \wedge \pi_{(1)} \models_{39}^{16} \neg r\mathbf{U}t \wedge \mathbf{F}(p \wedge \mathbf{F}c)$	(*by* $\dot{1}$)
$\Leftrightarrow$	$\pi_{(1)} \models_{39}^{16} \neg r\mathbf{U}t \wedge \pi_{(1)} \models_{39}^{16} \mathbf{F}(p \wedge \mathbf{F}c)$	
iff	$(\forall n.17 \leq n \leq 22 \wedge \pi_{(1)} \models_{39}^{n} \neg r) \wedge$ $\pi_{(1)} \models_{39}^{23} t \wedge \pi_{(1)} \models_{39}^{16} \mathbf{F}(p \wedge \mathbf{F}c)$	(*by* $\dot{8}$)
$\Leftrightarrow$	$(\pi_{(1)} \models_{39}^{17} \neg r) \wedge (\pi_{(1)} \models_{39}^{18} \neg r) \wedge \cdots (\pi_{(1)} \models_{39}^{22} \neg r)$ $\wedge \pi_{(1)} \models_{39}^{23} t \wedge \pi_{(1)} \models_{39}^{16} \mathbf{F}(p \wedge \mathbf{F}c)$	
iff	$\mathbf{True} \wedge \mathbf{True} \wedge \cdots \mathbf{True}$ $\wedge\, \mathbf{True} \wedge \pi_{(1)} \models_{39}^{16} \mathbf{F}(p \wedge \mathbf{F}c)$	
$\Leftrightarrow$	$\pi_{(1)} \models_{39}^{26} p \wedge \mathbf{F}c$	
iff	$\pi_{(1)} \models_{39}^{26} p \wedge \pi_{(1)} \models_{39}^{26} \mathbf{F}c$	(*by* $\dot{3}$)
iff	$\mathbf{True} \wedge \pi_{(1)} \models_{39}^{26} \mathbf{F}c$	(*by* $\dot{1}$)
$\Leftrightarrow$	$\pi_{(1)} \models_{39}^{26} \mathbf{F}c$	
iff	$\pi_{(1)} \models_{39}^{39} c$	(*by* $\dot{6}$)
iff	**True**	(*by* $\dot{1}$)

For satisfiability checking of $\pi_{(1)} \models$ (iii), the proposition represented by the satisfaction relation is on the top and is the starting point of the process in Table 4.

For the first step, we apply rule $\dot{6}$ in Table 3 to the proposition, and obtain second line of the process. The rule is defined for the propositions in the form $\pi \models_k^i \mathbf{F}\phi$. The rule can be applied, since the proposition on the top is in this form. (To clarify the application of this rule in this step, $\dot{6}$ is noted on the right of the second line of the process.) The rule $\dot{6}$ states that propositions in the form $\pi \models_k^i \mathbf{F}\phi$ holds, iff there exists some j between the current position (i) and the given cost bound (k) such that $\pi \models_k^j \phi$ holds. This application renders the substitution of $i = 0$, $k = 39$, and $\phi = o \wedge (\neg r\mathbf{U}t) \wedge \mathbf{F}(p \wedge \mathbf{F}c)$. Accordingly, we need to find some j such that $0 \leq j \leq 39$ and that $\pi_{(1)} \models_{39}^j o \wedge \neg r\mathbf{U}t \wedge \mathbf{F}(p \wedge \mathbf{F}c)$. For the only possible interpretation for this, $j = 16$ is assumed since truck route (1) visits location o only at the cost of 16 on its route.

In the second step to proceed from the second to the third line of the process, we apply rule $\dot{3}$ in Table 3. The rule is defined for the propositions in the form $\pi \models_k^i \phi_1 \wedge \phi_2$. This rule is applicable here, since the proposition in the second line is in this form. This application renders the two propositions of $\pi_{(1)} \models_{39}^{16} o$ and $\pi_{(1)} \models_{39}^{16} \neg r\mathbf{U}t \wedge \mathbf{F}(p \wedge \mathbf{F}c)$.

In the third step, the process proceeds from the third to the fourth line by applying rule $\dot{1}$ in Table 3 to a sub-proposition $\pi_{(1)} \models_{39}^{16}$ in the third line. The rule is defined for the propositions in the form $\pi \models_k^i l$. And it is applicable here, since the sub-proposition $\pi_{(1)} \models_{39}^{16} o$ is in this form. This application renders **True** since $\pi_{(1)}(16) = o$; hence the fourth line is obtained in the process.

Next, the process proceeds from forth to fifth line just as a consequence of applying the logical equivalence rule of "**True** $\wedge$ p $\Leftrightarrow$ p, where p stands for any propositional formula. Note that in the process in Table 4, the application of a logical equivalence rule is clarified using the symbol $\Leftrightarrow$, to explicitly distinguish it from the application of one of the rules in Table 3 when such a logical equivalence rule is applied.

The process further proceeds in a similar way by iteratively applying either one of the rules in Table 3 or a logical equivalence rule. And as it is shown, the process in the end lands in **True**, which means satisfiability checking of $\pi_{(1)} \models$ (i) is evaluated as **True**. This concludes truck route $\pi_{(1)}$ satisfies delivery requirement (i) with the cost of 39.

Though in above it is only shown that truck route $\pi_{(1)}$ satisfies delivery requirement (i) within the cost bound of 39 by satisfiability checking of $\pi_{(1)} \models$ (i), it can be also shown that $\pi_{(1)}$ satisfies all of the delivery requirements (i), (ii) and (iii) with the cost of 39; i.e., $\pi_{(1)} \models_{39}$ (i) $\wedge$ (ii) $\wedge$ (iii). On the other hand, truck route (2) fails to satisfy all of the delivery requirements (i), (ii) and (iii) within the cost bound of 39; i.e., $\pi_{(2)} \not\models_{39}$ (i) $\wedge$ (ii) $\wedge$ (iii) where $\pi_{(2)}$ denote truck route (2). But also, if the allowed cost is extended, for example 42, the $\pi_{(2)}$ is able to satisfy the all of the delivery requirements (i), (ii) and (iii); i.e., that means $\pi_{(2)} \models_{42}$ (i) $\wedge$ (ii) $\wedge$ (iii).

Secondly, the semantics w.r.t. a route map, defined in Definition 4, is demonstrated. The semantics w.r.t. defines the interpretation of a delivery requirement (ϕ) w.r.t. a route map (M) and given cost bound (k). The interpretation is made by evaluating a satisfaction relation of a route map to a delivery requirement within a cost bound to **True** (written as $M \models_k \phi$) or to **False** (written as $M \not\models_k \phi$), which respectively means the route map M satisfies the delivery requirement ϕ or fails to satisfy it within a given cost bound k. According to Definition 4, the route map (M_1) in Fig. 1 satisfies the delivery requirement (i), (ii) and (iii) with a cost bound of 39, i.e., $M_1 \models_{39}$ (i) $\wedge$ (ii) $\wedge$ (iii), since a truck route (1) of the route map satisfies the requirement within the cost bound. On the other hand, M_1 fails to satisfy it, since there is no truck route of the route map satisfies the requirement within the cost bound.

4.3 Optimal Truck Routes

Optimal truck routes, which are paths satisfying given delivery requirements with the minimum cost, are defined as:

Definition 5 (Optimal truck routes): A path π of a route map M is an optimal path that satisfies a delivery requirement ϕ if it satisfies the following conditions:

1. $\pi \models_i \phi$ for some i, and
2. if $\pi' \models_j \phi$ for all π' in M and for some j then $i \leq j$.

Example 1 (Optimal truck routes): Take the example in Sect. 2. As already mentioned, there are several truck routes of the route map that satisfy the given delivery requirements such as (i) and (ii). And among them truck routes (1) is an optimal path in terms of Definition 5.

5. Implementing an Automated Route Planner for the Milk-Run Logistics Framework Using the NuSMV Model Checker

This section explains a prototype system, which we have implemented, of the milk-run logistics framework given in the previous sections. The system, given a route map and delivery requirements (specified by LTL), automatically finds an optimal truck route that satisfies the requirements based on the framework. We use the NuSMV model checker [10], [11] to realize such a system. Several gaps to realize the system by NuSMV, and techniques to bridge them are explained. Also we show experimental results of the implementation as its evaluation.

5.1 Encoding Route Maps into NuSMV Codes

In applying NuSMV to realize a system for the framework, the route maps are encoded into NuSMV codes. In encoding route maps to NuSMV codes, one gap we need to bridge is how to encode the weights of edges in route maps in NuSMV codes so that the model checking algorithm of NuSMV can find optimal paths.

We realize this by introducing extra nodes, which we call *dummy nodes*, to express the weights. Consider, for example, a route (i.e., edge) connecting two locations with weight "3". Figure 2 shows how the weight of each route in the route map (on the left) is expressed in the NuSMV codes using two dummy nodes of _ef1 and _ef2 (on the right). That is, the weight of n of a route in the route map is expressed by preparing $n-1$ dummy nodes between the two locations. Table 5 shows actual NuSMV codes for the map shown in Fig. 1.

Also, note that, due to this implementation design, the *neXt* operator ($\mathbf{X}\phi$) is excluded from the LTL language set, which is used for specifying delivery requirements, in this implementation.

5.2 Bridging Gaps between UMC and EMC

The second gap lies between EMC, which our framework assumes, and UMC, which NuSMV is based on [10], [11]. In EMC, only one path of a model that satisfies a formula is required to prove the satisfaction relation between the model and the formula, which we may call the "witness". In UMC, which NuSMV assumes, it is required that all the paths of a model satisfy a given formula, for the satisfaction relation of

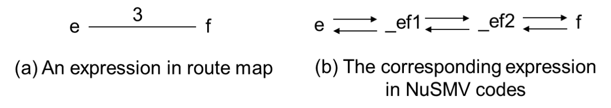

e
3
f
(a) An expression in route map
e
_ef1
_ef2
f
(b) The corresponding expression in NuSMV codes

Fig. 2 How to encode weights of route map in NuSMV codes.

Table 5 NuSMV code for route map of Fig. 1.

```
Module main
VAR
 node : {a, _ab3, _ab2, _ab1, b, _ac1, c, ..., _dummy};
ASSIGN
 init(node) := a;
 next(node) := case
       node = a : {_ab3, _ac1, _ad1};
       node = _ab3 : {a, _ab2};
       node = _ab2 : {_ab3, _ab1};
       node = _ab1 : {_ab2, b};
       node = _ac1 : {a, c};
       node = _ad1 : {a, d};
       node = b : {_ab1, _be1, _bd2, _bf2, _bg2};
       node = _be1 : {b, e};
       node = _bd2 : {b, _bd1};
       node = _bd1 : {_bd2, d};
       node = c : {_ac1, _cd1, _ci2};
--          continue ...

```

the model to the formula to hold. That is, the existence of a path that fails to satisfy the formula violates the satisfaction relation, which is a so-called *counter-example*.

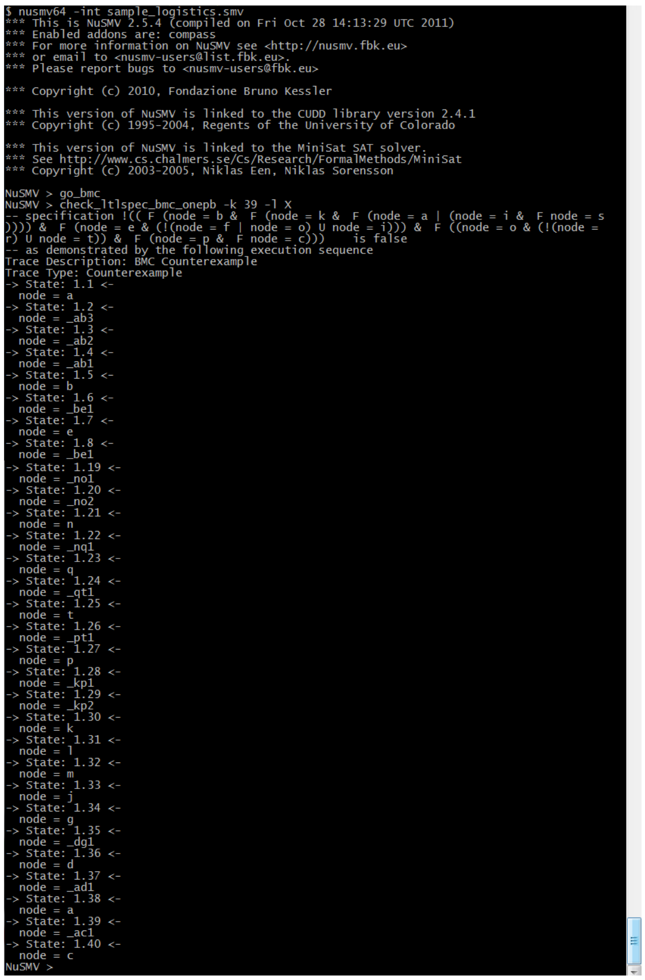
Here, an important observation for bridging the gap is that due to the semantics design, which keeps duality in the bounded semantics setting in Definition 3, finding a witness in EMC corresponds to finding a counter-example to the negation form of the formula in UMC. Accordingly, to build an automated route planner for the framework using NuSMV, it suffices to apply the negation form of the LTL formula to NuSMV, and in this way a generated counter-example is a witness and hence is a truck route that satisfies the delivery requirements.

5.3 Finding Optimal Paths Using NuSMV

The basic idea to find an optimal truck route using NuSMV is to use the BMC function of NuSMV. BMC, which is a model checking based on bounded semantics, can find a counter-example that violates a given formula, with the shortest length due to the breadth-first nature of SAT search procedures. Also Definition 5 defines the optimal truck routes as the ones which satisfy a given delivery requirements with the minimum weights. Together with the technique to bridge UMC and EMC discussed previously, we can find an optimal truck route that satisfies given delivery requirements.

In realizing this idea using NuSMV, instead of the simple and standard command "bmc", we use the "check_ltlspec_bmc_onepb" command with providing natural numbers k to the command to specify the length of counter-examples (hence the length of truck route), and "X" for the option "l" to obtain paths with "no loop-back". The actual procedure of finding an optimal using NuSMV and its response from NuSMV is demonstrated in Fig. 3.

This elaborated procedures, instead of the simple "bmc" command, is required in order to avoid that NuSMV finds a truck route which does not comply with the notion



```
$ nusmv64 -int sample_logistics.smv
*** This is NuSMV 2.5.4 (compiled on Fri Oct 28 14:13:29 UTC 2011)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

NuSMV > go_bmc
NuSMV > check_ltlspec_bmc_onepb -k 39 -l X
-- specification !(( F (node = b &  F (node = k &  F (node = a | (node = i &  F node = s
)))) &  F (node = e & (!(node = f | node = o) U node = i))) &  F ((node = o & (!(node =
r) U node = t)) &  F (node = p &  F node = c)))    is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  node = a
-> State: 1.2 <-
  node = _ab3
-> State: 1.3 <-
  node = _ab2
-> State: 1.4 <-
  node = _ab1
-> State: 1.5 <-
  node = b
-> State: 1.6 <-
  node = _be1
-> State: 1.7 <-
  node = e
-> State: 1.8 <-
  node = _be1
-> State: 1.19 <-
  node = _no1
-> State: 1.20 <-
  node = _no2
-> State: 1.21 <-
  node = n
-> State: 1.22 <-
  node = _nq1
-> State: 1.23 <-
  node = q
-> State: 1.24 <-
  node = _qt1
-> State: 1.25 <-
  node = t
-> State: 1.26 <-
  node = _pt1
-> State: 1.27 <-
  node = p
-> State: 1.28 <-
  node = _kp1
-> State: 1.29 <-
  node = _kp2
-> State: 1.30 <-
  node = k
-> State: 1.31 <-
  node = l
-> State: 1.32 <-
  node = m
-> State: 1.33 <-
  node = j
-> State: 1.34 <-
  node = g
-> State: 1.35 <-
  node = _dg1
-> State: 1.36 <-
  node = d
-> State: 1.37 <-
  node = _ad1
-> State: 1.38 <-
  node = a
-> State: 1.39 <-
  node = _ac1
-> State: 1.40 <-
  node = c
NuSMV >
```

Fig. 3 A screen dump for finding a truck route with cost 39 w.r.t. the route map in Fig. 1 and delivery requirements (i) ∧ (ii) ∧ (iii) using NuSMV.

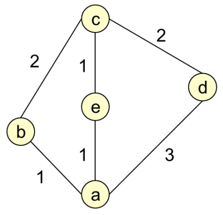


Fig. 4 A minimal example for the loop-back option in NuSMV.

of "optimal truck route" in Definition 5. Specifically, for example, consider the route map in Fig. 4 where a truck is on location b currently, and that the following delivery requirements are given:

- Truck visits location d, and then a, and then c; formally,

$$\mathbf{F}(d \wedge \mathbf{F}(a \wedge \mathbf{F}c))$$

According to the notion of optimal truck routes defined in Definition 5, an optimal route is as follows:

$$b^0 \to^2 c^2 \to^2 d^4 \to^3 a^7 \to^1 e^8 \to^1 c^9$$

But if we apply the simple procedure using the "bmc" command of NuSMV, NuSMV returns the result in finding the following truck route:

$$b^0 \rightarrow^2 c^2 \rightarrow^2 d^4 \rightarrow^3 a^7 \rightarrow^1 b^8$$

What kinds of result does NuSMV answer? It is a truck route containing "loop-back" (i.e., the first location and the last location b in the path is same), which satisfies the (negation of) delivery requirements. This truck route satisfies the delivery requirements if we consider it as a "cycle truck route", since it can be seen that after visiting location d and then a as required, the truck in the route will visit location c. This truck route can be regarded as one notion of a truck routes satisfying the delivery requirements, but not the one complying with the notion in Definition 5.

5.4 Computational Complexity of the Implementation Design

We analyze computational complexity of this implementation design for the framework. As mentioned in Sect. 3, the complexity of BMC using SAT is analyzed via the number of propositional variables to appear by SAT encoding, represented as $O(k \cdot |\log(S)| + (k+1)^2 \cdot |\phi|)$, where S is the number of states in model M, k is the bound, and $|\phi|$ is the length of ϕ. Since the implementation encodes route maps into Kripke models, here we need to analyze the number of states accompanied by the encoding. In encoding, dummy locations to express the weights of a route map as well as locations are encoded to the states of Kripke model. Hence the number of states of a Kripke model encoded from route map M, which we consider as the size of route map M and denote as $size(M)$, is given as the sum of the number of locations (S) and dummy locations; i.e., $size(M) = |Loc| + \sum\{n-1 \mid ((a,b),n) \in R\}$. That is, the number of states in the Kripke models due to the encoding increases in a linear manner; hence the number of propositional variables to appear in the encoded formula is $O(k \cdot |\log(size(M))| + (k+1)^2 \cdot |\phi|)$.

5.5 Experimental Results

As an experiment to demonstrate the feasibility of the implementation design, we show benchmark results based on the analysis of computational complexity above. From the complexity analysis, it is known that cost bounds (i.e., the length of paths) and the length of the LTL formula are the main contributors to the computational cost. Hence the experiment is designed based on interactions of these two factors where the size of route map is fixed at 200. Also, due to the advantageous property of the breadth-first search in BMC, we can apply model checking for each different cost bound on a different computing node in parallel. Thus, for cost bounds, we show results obtained by applying model checking with the exact cost bound k instead of all the cost bounds from 1 to k. Table 6 shows the experimental results, where each result shows the average execution times over five trials. We distinguish the executions that result in finding a path that satisfies a given requirement with a gray-coloured cell from those that result in failing with a white-coloured cell. We can observe that the experimental results almost conform to the complexity analysis; i.e., the computing cost w.r.t. the length of the formula and cost bound k increases in an exponential manner. Also, it is interesting to observe that the cost w.r.t. the bound k decreases after finding a path; this reflects the property of "phase transition", which is a characteristic to SAT problem well-investigated in [12], [13].

Table 6 Experimental results for finding optimal truck routing.

	The length of formula ϕ $length(\phi)$					
cost bound	20	30	40	50	60	70
$k = 20$	4s	3s	4s	3s	4s	4s
$k = 30$	6s	14s	68s	66s	2m40s	2m39s
$k = 40$	5s	16s	2m52s	21m49s	122m0s	277m58s
$k = 50$	9s	72s	3m34s	5m22s	80m0s	136m23s
$k = 60$	11s	47s	4m26s	6m38s	18m54s	128m18s

# The experiments were conducted on a machine with an AMD Dual-Core Opteron 2220 CPU @2.8GHz, 33GB of RAM and Debian Linux 5.0.10.

6. Evaluation

This section is devoted to evaluate our framework and its prototype for milk-run transport logistics. The evaluation is conducted, from practical viewpoints, on the two main contributions of the paper. One is on the specification language for expressing delivery requirements for milk-run logistics operations deviced in the framework. And the other is on the efficiency aspect of the prototype of the framework we have implemented in Sect. 5.

6.1 Evaluation of the Specification Language

Our framework adopts LTL as a specification language to express delivery requirements (and hence, truck routes), for milk-run logistics operations. We evaluate this approach by way of the framework developed by Satoh in [1], [2], which is in real use in logistics industries, deviced also with a specification language for expressing truck routes for the milk-run logistics operations.

"Satoh's language" in [1], [2] is developed in the context of his development of a foundational framework for "a truck selection system" for milk-run logistics operations. The language is designed based on CCS which is a process calculus, in order to flexibly expressing various complex orders of locations, assuming milk-run operations. The language design is so simple, since it abstracts various other aspects of the logistics away, aiming to develop a foundational framework. But the language is shown to be practically expressive for a real use in some logistics industries; the language reduces complexity of managing complex truck routes in milk-run logistics operations.

To claim practicality of adopting LTL as a specification language for milk-run logistics operations by way of

Satoh's language, we argue the expressive aspect of LTL w.r.t. Satoh's language. That is, we argue what Satoh's language can express and what LTL can express. Theoretically, Satoh's language is more expressive than LTL, by using the following known facts:

- Satoh's language is an extension of CCS [1], [2],
- CCS is as expressive as modal μ-calculus [14],
- modal μ-calculus is more expressive than CTL* [15]
- CTL* is more expressive than LTL, [16]

Especially what makes Satoh's language, characterized as modal μ-calculus, more expressive than LTL is mainly some CTL formulas, which embed the path quantifiers ($\forall\phi$) and hence cannot expressed, according to the following theorem in [16] (p.335):

Theorem 6.18 (p.335 in [16]) : [Criterion for Transforming CTL Formulae into Equivalent LTL Formulae] Let Φ be a CTL formula, and ϕ the LTL formula that is obtained by eliminating all path quantifiers in Φ. Then: $\Phi \equiv \phi$ or there does not exist any LTL formula that is equivalent to Φ.

But the absence of embedded path quantifiers in CTL formulas, which express all/some branches on a tree, does not spoil practicality of using LTL as a specification language to express truck route, especially for the purpose of route planning for milk-run logistics operations. This is because, as explained in Sect. 2.3, in our framework of route planing, we assume a truck route is a single path of route map and delivery requirements are interpreted in a single path like LTL, but not on a tree like CTL.

Also note that this paper focuses on and is confined to developing a framework of route planning in the context of milk-run logistics operations by applying model checking techniques. We leave more direct and thorough evaluation of practicality of the framework adopting LTL in real world logistics industries to our future work.

6.2 Evaluation of the Efficiency of the Implemented Automated Route Planner

Today, milk-run operations in transport logistics are used in various industries such as food, automobile manufacturing, military, as well as the dairy industry. Also operation methods also differ depending on their purposes, including the following examples:

- Small route maps are used in some operations, but large ones need to be used in others.
- The number of locations which a truck visit is large in some operations, but not so in the other operations.
- The computing time for finding truck routes is strict in some operations but is not so in the others.

According to the experimental results shown in Sect. 5.5, obviously the prototype implementation does not suit such operations with large route maps, strict computing time, and complex delivery requirements from the view point of efficiency.

But still, there are some settings in logistics industries, which the current implementation of the prototype with this efficiency can be applied to. One of such settings is in dairy industry. Generally, in dairy industry where milk-run logistics operations are often used, a truck with dairy goods can not travel for long time, to avoid affecting their quality. Therefore, in such settings, both of route maps dealt with and the number of locations which a truck visits should not be large. More specifically, according to an expert on this logistics domain, in such settings the number of locations that a truck should visit in such settings is 5 to 10 locations at most, which approximately corresponds to 15 to 30 in the length of the formula respectively. The current prototype of the efficiency as shown in Table 6 is applicable in such a setting, since it works better way than hand-made planning as we can automatically obtain truck routes which accurately satisfy complex delivery requirements in a faster way.

Also, note the main focus of this paper is to develop a foundational framework for route planning, where complex delivery requirements are flexibly specified and a mechanism for finding truck routes to satisfy such complex delivery requirements is deviced, and to implement a prototype of the framework. The efficiency aspect will be tackled on in our future work, by improving the algorithms for route planning in our setting of milk-run logistics operations.

7. Related Work

As mentioned, this work is inspired by [1], [2] to provide a formal framework for milk-run transport logistics which reduces its complexity. But our work differs from those in its purpose; i.e., the purpose of our work is to develop a formal framework for automated route planning, while that of [1], [2] is to develop a framework for a mechanism for truck selection. This makes differences in the technologies used in the frameworks. Our framework uses model checking techniques, while [1], [2] uses bisimulation checking techniques. Also, this work is the first to apply model checking technologies, originally for automated verification, to automated route planning for milk-run transport logistics.

Various shortest-path problems and their algorithms in graph theory, such as "single-source shortest path problems" (and Dijkstra's algorithm), "all-pairs shortest path problems", "travelling salesman problems", "widest path problems", etc., share a common link with our framework in finding optimal paths provided weighted graphs. Note that these graph algorithms are designed to solve their own specific and fixed problems; e.g., Dijkstra's algorithm is designed specifically for a "single-source shortest path problem". On the other hand, this work proposes a general framework; i.e., it provides a way to flexibly specify various problems as "delivery requirements" by LTL, and a mechanism to solve these problems by automatically finding shortest paths for the given problems in a unified way.

8. Conclusion and Future Work

Conclusion

In this paper, we have developed an automated route planning framework for milk-run transport logistics, which given a route map and delivery requirements finds optimal truck routes that satisfy requirements with the lowest cost. The framework is realized by applying model-checking techniques. It uses LTL as a specification language for specifying delivery requirements in milk-run logistics, which are complex w.r.t. the order of locations which trucks should visit. We have discussed the framework formally, including the notion of "optimal truck routes", by applying bounded semantics. Further based on the formal foundation we implemented the framework by applying the NuSMV model checker. As the milk-run logistics is used in various ways in various industries, we have shown the computational complexity and experimental results as the feasibility of the implementation.

Future Work

This paper focuses on developing the framework and on demonstrating its feasibility, by implementing the system for the framework in a simple manner as an early trial; i.e., we leave the aspect of efficiency and correctness of the system design and implementation to our future papers. Another important research direction is to develop a Domain Specific Language (DSL) for specifying delivery requirements in milk-run logistics. In this paper, LTL is used for that purpose in order to show the applicability of model checking to a route planning problem in milk-run logistics. However more suitable languages can be designed to more effectively express delivery requirements in milk-run logistics in practice.

Acknowledgement

The authors would like to thank Makoto Takeyama in Kanagawa University for his time and insightful suggestions throughout this research.

References

[1] I. Satoh, "A specification framework for earth-friendly logistics," Proc. 28th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE), LNCS, vol.5048, pp.251–266, 2008.
[2] I. Satoh, "A formal approach for milk-run transport logistics," IEICE Trans. Fundamentals, vol.E91-A, no.11, pp.3261–3268, Nov. 2008.
[3] R. Milner, Communication and concurrency, PHI Series in computer science, Prentice Hall, 1989.
[4] A. Biere, A. Cimatti, E.M. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," Proc. 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS), LNCS, vol.5048, pp.193–207, 1999.
[5] A. Biere, A. Cimatti, E.M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," Advances in Computers, vol.58, pp.118–149, 2003.
[6] T. Kitamura and K. Okamoto, "Automated route planning for milk-run transport logistics," Proc. 4th International Workshop on Parallel and Distributed Algorithms and Applications (PDAA), 2012.
[7] E.W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol.1, pp.269–271, 1959.
[8] E. Clarke, D. Kroening, J. Ouaknine, and O. Strichman, "Completeness and complexity of bounded model checking," Proc. 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI), LNCS, vol.2937, pp.85–96, 2004.
[9] E. Clarke, D. Kroening, J. Ouaknine, and O. Strichman, "Computational challenges in bounded model checking," STTT, vol.7, no.2, pp.174–183, 2005.
[10] "NuSMV: A new symbolic model checker," available from http://nusmv.fbk.eu/.
[11] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV Version 2: An OpenSource tool for symbolic model checking," Proc. 14th International Conference on Computer Aided Verification (CAV), LNCS, vol.2404, pp.359–364, 2002.
[12] P. Cheeseman, B. Kanefsky, and W.M. Taylor, "Where the really hard problems are," Proc. 12th International Joint Conference on Artificial Intelligence (IJCAI), pp.331–337, 1991.
[13] D. Mitchell, B. Selman, and H. Levesque, "Hard and easy distributions of SAT problems," Proc. 10th National Conference on Artificial Intelligence (AAAI), pp.459–465, 1992.
[14] C. Stirling, Modal and Temporal Properties of Processes, Springer, 2001.
[15] M. Dam, "CTL* and ECTL* as fragments of the modal mu-calculus," Theor. Comput. Sci., vol.126, no.1, pp.77–96, 1994.
[16] C. Baier and J.P. Katoen, Principles of model checking, MIT Press, 2008.

Takashi Kitamura is currently a researcher in National Institute of Advanced Industrial Science and Technology in Japan. He received the B.S. and M.S. degrees in informatics from Shizuoka University, Japan, in 2001 and 2003, respectively. He received the doctor degree of Engineering (Computer Software and Theory) from Institute of Software, Chinese Academy of Sciences in Beijing in 2008.

Keishi Okamoto received his B. Science, M. Science and Doctor of Science degrees in Mathematics from Waseda University, Japan in 2008. From 1998 to 2001, he was a research associate in Waseda University. He was a post-doctoral researcher from 2004 to 2006, a technical staff from 2006 to 2008, an invited senior research scientist from 2008 to 2011 in National Institute of Advanced Industrial Science and Technology (AIST). Since 2011, he has been an associate professor in Sendai National College of Technology. His current research interests include mathematical logic and formal methods. He is a member of two learned societies, Japan Society for Software Science and Technology (JSSST) and Mathematical Society of Japan (MSJ).