Symbolic Model Checking of Finite Precision Timed Automata *

Rongjie Yan[1,2], Guangyuan Li[1], and Zhisong Tang[1]

[1] Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences,Beijing,100080,China
[2] Graduate School of the Chinese Academy of Sciences,Beijing,100039,China
{yrj,ligy}@ios.ac.cn

Abstract. This paper introduces the notion of finite precision timed automata (FPTAs) and proposes a data structure to represent its symbolic states. To reduce the state space, FPTAs only record the integer values of clock variables together with the order of their most recent resets. We provide constraints under which the reachability checking of a timed automaton can be reduced to that of the corresponding FPTA, and then present an algorithm for reachability analysis. Finally, the paper reports some preliminary experimental results, and analyzes the advantages and disadvantages of the new data structure.

Key words: finite precision timed automata, model checking, symbolic methods

1 Introduction

Timed automata (TAs) [1] provide a formal framework for the automatic analysis and verification of real-time systems, and in the past few years several tools for the model checking of TAs have been developed and used, including Uppaal [20], Kronos [15, 11], Red [22–24] and Rabbit [9].

State space explosion is likely to be the most serious problem that any model checker has to deal with. Within the model checking community, there were many different attempts to reduce memory consumption and to accelerate the speed of exploration, including symmetry reduction [10, 18], partial order reduction [6] and active clock reduction [16] (all based on the structural characteristics of the system being verified), as well as region equivalence partition of state space [1], and the discretization of time models [3, 9].

In addition to the approaches mentioned above, many works were based on symbolic representations of the state space. The region equivalence of [1] is the precursor of the symbolic methods in which the state space is covered using regions with the same integer parts of clock values and the ordering of

* Supported by 973 Program of China under Grant No. 2002cb312200; and the National Natural Science Foundation of China under Grant Nos. 60273025, 60223005, 60421001.

fractional parts. As a result, an infinite state space may then become finite. A zone, based on region equivalence, uses a set of clock difference constraints to represent all the states satisfying these constraints. Currently, most of verifiers, like Uppaal, Kronos, and Red, use zones to represent symbolic states. And there exist different data structures to describe the constraint sets, for example, DBM (difference bound matrices) [5] and BDD-like (binary decision diagrams) [13] data structures. In DBMs, employed by Uppaal and Kronos, a constraint set is expressed as a weighted, directed graph with vertices corresponding to all the clock variables and a zero-vertex 0. In BDD-like data structures, a node of the decision tree represents a clock difference, an edge is labelled with an integral interval, and a node together with an outgoing edge represents a constraint. Uppaal and Kronos implemented such data structures too [4, 12].

Discrete timed automata [2] may achieve higher efficiency in analysis and verification due to having fewer states, but they are not suitable to describe asynchronous systems. On the other hand, timed automata with continuous semantics [1] are appropriate to both synchronous and asynchronous systems, but the complexity remains very high. To apply the mature techniques in discrete time models to improve the efficiency of model checking, we try to discretize the continuous time.

The present paper introduces a finite precision timed automaton (FPTA) together with a data structure, different from DBMs and BDDs, for a symbolic representation of the state space (note that *Finite precision* means that the clocks have integral valuations). It can be shown that the reachability problem of TA may be reduced to that of the corresponding FPTA under certain constraints (see Theorm 1). The paper also develops a reachability analysis algorithm for the FPTAs, and shows some initial experimental results.

The paper is organized as follows. In section 2, we briefly recall the definition of TAs and their semantics. In section 3, we introduce FPTAs. In section 4, we discuss the reachability problem equivalence for the TAs and FPTAs. In section 5, we present the new symbolic data structure and the reachability analysis algorithm for the FPTAs. In section 6, we give some experiment results. In the concluding section, we discuss related work.

2 Timed Automata

A timed automaton (TA), proposed by Alur and Dill [1], is a finite state automaton extended with a finite set of real-valued clock variables. Nodes of a TA represent locations, and arcs represent transitions between them. Clock constraints within a node (invariants) restrict the time that can elapse in it. Constraints labelling arcs act as guards for transitions between the nodes.

Definition 1. *(Syntax of Timed Automata). Let X be a finite set of clocks, and $C(X)$ be the clock constraint set over X, given by the syntax:*

$$\phi \ ::= \ (x \sim c) \mid \phi_1 \wedge \phi_2 \mid true$$

where $x \in X$, $\sim \in \{<, \leq, >, \geq\}$ and $c \in \mathbb{N}^+$ ($\mathbb{N}^+$ is the set of non-negative integers).

A timed automaton *over X is a tuple $A = \langle L, l_0, \Sigma, X, I, E \rangle$, where*

- *L is a finite set of locations, and $l_0 \in L$ is the initial location,*
- *I is a mapping that labels each location $l \in L$ with some constraint in $C(X)$,and $I(l)$ is called the invariant of l,*
- *Σ is a finite set of synchronization labels, and*
- *$E \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ is the set of transitions.*

A transition $(l, \phi, \sigma, Y, l') \in E$ means that one can move from the location l to l' through a transition labelled with $\sigma \in \Sigma$. Moreover, ϕ the guard must be satisfied by the current clock values, and all the clocks in Y ($Y \subseteq X$) are reset to 0.

A clock valuation is a function $\mu : X \mapsto \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of non-negative reals. μ_X denotes the set of all clock valuations over X. For $t \in \mathbb{R}^+$, $\mu + t$ denotes the clock valuation such that $\mu(x + t) = \mu(x) + t$, for all $x \in X$. For $Y \subseteq X$, $\mu[Y := 0]$ denotes the clock valuation such that $\mu[Y := 0](x) = 0$, for all $x \in Y$ and otherwise $\mu[Y := 0](x) = \mu(x)$. μ satisfies a constraint $\phi \in C(X)$, denoted by $\mu \models \phi$, if ϕ evaluates to *true* under the assignment given by μ.

The continuous semantics of a timed automaton $A = \langle L, l_0, \Sigma, X, I, E \rangle$ over X is defined as a transition system $[\![A]\!]_C = \langle S, s_0, \Sigma \cup \mathbb{R}^+, \rightarrow \rangle$, where $S = L \times \mu_X$; $s_0 = (l_0, \mu_0)$ is the initial state where $\mu_0(x) = 0$ for all $x \in X$; and the transition relation $\rightarrow$ comprises two kinds of moves:

- delay transition: $(l, \mu) \xrightarrow{\delta} (l, \mu + \delta)$, if $\delta \in \mathbb{R}^+$ and $\mu \models I(l)$ and $\mu + \delta \models I(l)$;
- discrete transition: $(l, \mu) \xrightarrow{\sigma} (l', \mu[Y := 0])$, if $(l, \phi, \sigma, Y, l') \in E$ and $\mu \models \phi$ and $\mu[Y := 0] \models I(l')$.

In the transition system of $A = \langle L, l_0, \Sigma, X, I, E \rangle$, for a state $s_k = (l, \mu)$ where $l \in L$, if there exists a transition sequence such that $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{k-1}} s_k$, then s_k is called reachable in the continuous semantics of A where $\alpha_i \in \Sigma \cup \mathbb{R}^+$. Given a location l and a clock constraint ϕ, if there exists a reachable state (l, μ) such that $\mu \models \phi$, then (l, ϕ) is called reachable in A.

3 Finite Precision Timed Automata

The syntax of FPTAs is the same as that of TAs. The feature that differentiates an FPTA from a discrete time TA is that it can distinguish the ordering of clock resets, by the introduction of an *order*. To define the semantics of FPTAs, we introduce the notion of *order* firstly.

Definition 2. *(Order). An order over X is a mapping $o : X \mapsto \mathbb{N}^+$, and the set of all such mappings is denoted by o_X. For $x_1, x_2 \in X$ with $o(x_1) < o(x_2)$, we say that the order of x_1 is less than that of x_2.*

Since different order valuations may represent the same ordering relationship between the clocks, we introduce order normalization.

Definition 3. *(Order Normalization).*

1. *An order o over X is normalized if the image $o(X)$ is an initial interval of $\mathbb{N}^+$.*
2. *Two orders, o and o', are equivalent if $o(x) \leq o(y)$ iff $o'(x) \leq o'(y)$, for all $x, y \in X$.*
3. *For each order o, we denote by $norm(o)$ the unique normalized order which is equivalent to o.*

For example, the normalization of the order $o_1 = (1, 0, 3, 4, 3)$ is $norm(o_1) = (1, 0, 2, 3, 2)$.

For an FPTA $A = \langle L, l_0, \Sigma, X, I, E \rangle$, a clock valuation is a mapping $v : X \mapsto \mathbb{N}^+$, and the set of all such valuations is denoted by v_X. A state of A is $s = (l, (v, o))$ where $l \in L$, $v \in v_X$ is a clock valuation, $o \in o_X$ is a normalized order, and (v, o) is the clock information of the state. For a state $s = (l, (v, o))$ and a constraint $\phi \in C(X)$, if $v \models \phi$, we say that s satisfies (l, ϕ), denoted by $s \models (l, \phi)$.

The semantics of A is the transition system $[\![A]\!]_{FP} = \langle S, s_0, \Sigma, \rightarrow \rangle$, where $S = L \times (v_X \times o_X)$; $s_0 = (l_0, (v_0, o_0))$ is the initial state where $v_0(x) = o_0(x) = 0$ for all $x \in X$; and the transition relation $\rightarrow$ comprises two kinds of moves:

- delay transition: $(l, (v, o)) \xrightarrow{\epsilon} (l, (v, o) \oplus k)$ (or simply $(l, (v, o)) \rightarrow (l, (v, o) \oplus k)$), if $k \in \mathbb{N}^+$ and $(v, o) \oplus k \models I(l)$, where $((v, o) \oplus k)(x) = (v(x) + (o(x) + k) \text{ div } m, (o(x) + k) \text{ mod } m)$ for $m = 1 + \max o(X)$;
- discrete transition: $(l, (v, o)) \xrightarrow{\sigma} (l', (v', o'))$, if there exists $(l, \phi, \sigma, Y, l') \in E$ such that $v \models \phi$ and $v' \models I(l')$ and $(v', o') = reset((v, o), Y)$ where $\text{reset}((v, o), Y) = (v[Y := 0], \text{norm}((o + 1)[Y := 0]))$ and for each $x \in X$:

$$v[Y := 0](x) = \begin{cases} 0 & \text{if } x \in Y \\ v(x) & \text{if } x \in X - Y, \end{cases}$$
$$(o + 1)[Y := 0](x) = \begin{cases} 0 & \text{if } x \in Y \\ o(x) + 1 & \text{if } x \in X - Y. \end{cases}$$

A run of an FPTA A is an infinite sequence of its discrete and delay transitions:

$$(l_0, (v_0, o_0)) \xrightarrow{\alpha_0} (l_1, (v_1, o_1)) \xrightarrow{\alpha_1} (l_2, (v_2, o_2)) \xrightarrow{\alpha_2} (l_3, (v_3, o_3)) \xrightarrow{\alpha_3} \ldots$$

where $\alpha \in \Sigma \cup \{\epsilon\}$.

Here we show one of runs for the FPTA in Figure 1. For convenience, in Figure 1, the discrete transition from l_0 to l_1 is called t_1; the other is called t_2.

Example 1. One of the runs for the FPTA in Figure 1 is:[1] $(l_0, 0000) \rightarrow (l_0, 1100) \rightarrow (l_0, 2200) \xrightarrow{a} (l_1, 2010) \rightarrow (l_1, 3001) \rightarrow (l_1, 3110) \rightarrow (l_1, 4101) \rightarrow (l_1, 4210) \rightarrow (l_1, 5201) \rightarrow (l_1, 5310) \xrightarrow{b} (l_0, 0301) \rightarrow \cdots$

[1] To simplify the representation of clock information, we list them as clock values followed by clock orders, e.g., the first two values of 0000 are the values of clock variable x and y, the last two values are their orders.

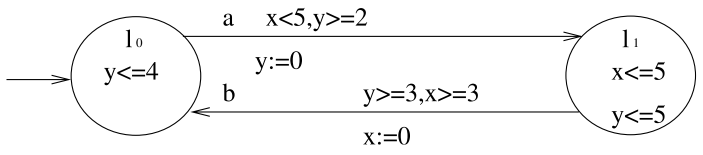

Fig. 1. A simple timed automaton

Before discrete transitions, all the clock values will increase by one after every time unit and the orders will not change. When the state $(l_0, 2200)$ is generated, the guard of t_1 is satisfied, and its occurrence leads to $(l_1, 2010)$. Now the guard of t_2 is not satisfied, and time elapses in location l_1 on condition that its invariant is satisfied. When $(l_1, 5310)$ is generated, transition t_2 can occur and generate successive states. □

Since safety and bounded liveness properties can be expressed in terms of reachability [8], many model checkers for real-time systems concentrate on the latter. The reachability checker of FPTAs will analyze whether (l, ϕ) is reachable.

Definition 4. *(The Reachable State of FPTAs). Let A be an FPTA. For a state $(l_n, (v_n, o_n))$, if there is a finite state sequence such that*

$$(l_0, (v_0, o_0)) \xrightarrow{\alpha_0} (l_1, (v_1, o_1)) \xrightarrow{\alpha_1} (l_2, (v_2, o_2)) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{n-1}} (l_n, (v_n, o_n))$$

then $(l_n, (v_n, o_n))$ is called reachable. Given a location l and a clock constraint ϕ, if there exists a reachable state $(l, (v, o))$ in A such that $v \models \phi$, we say that (l, ϕ) is reachable in A.

4 The relationship between FPTAs and TAs

In this section, we investigate the relationship between reachability in FPTAs and TAs. We have left proofs of Lemma 1 and 2 to Appendix A.

Definition 5. *(Left Closed and Right Open Timed Automata). A clock constraint ϕ generated by the syntax*

$$\phi \ ::= \ x \geq c \mid x < c \mid \phi_1 \wedge \phi_2 \mid true$$

is called left-closed and right-open (lcro-constraint). If all the constraints of a TA are of this kind, we call it an lcro-TA.

With the lcro-constraint ϕ, the clock valuation of TAs $\mu \models \phi$ iff $\lfloor \mu \rfloor \models \phi$ (where $\lfloor \mu \rfloor$ is the mapping which assigns every $x \in X$ an integer $\lfloor \mu(x) \rfloor$). Let A be an lcro-TA, we will investigate the relationship between the continuous semantics $[\![A]\!]_C$ and the finite precision semantics $[\![A]\!]_{FP}$.

Definition 6. *(Relation $\triangleright$). Let $\mu : X \mapsto \mathbb{R}^+$ be a clock valuation of TAs, and (v, o) be a clock information of FPTAs. Then $\mu \triangleright (v, o)$ if, for all $x, y \in X$, $v(x) = \lfloor \mu(x) \rfloor$ and $(frac(\mu(x)) < frac(\mu(y)) \Rightarrow o(x) < o(y))$, where $frac(r)$ denotes the fractional part of a non-negative real r.*

Lemma 1. *For an lcro-TA A, if (l, μ) is reachable in $[\![A]\!]_C$, then there exists (v, o) such that $(l, (v, o))$ is reachable in $[\![A]\!]_{FP}$, and $\mu \triangleright (v, o)$.*

Definition 7. *(Relation $\doteq$). Let $\mu : X \mapsto \mathbb{R}^+$ be a clock valuation of TAs, and (v, o) be a clock information of FPTAs. Then $\mu \doteq (v, o)$ if, for all $x, y \in X$, $v(x) = \lfloor \mu(x) \rfloor$ and $(frac(\mu(x)) < frac(\mu(y)) \Leftrightarrow o(x) < o(y))$.*

Lemma 2. *For an lcro-TA A, if $(l, (v, o))$ is reachable in $[\![A]\!]_{FP}$, then there exists μ such that (l, μ) is reachable in $[\![A]\!]_C$, and $(v, o) \doteq \mu$.*

Let $Reach_C(A) = \{(l, \phi) | l \in L, \phi \in C(X)$, and (l, ϕ) is reachable in the continuous semantics$\}$, and $Reach_{FP}(A) = \{(l, \phi) | l \in L, \phi \in C(X)$, and (l, ϕ) is reachable in the finite precision semantics$\}$.

Theorem 1. *Let A be an lcro-TA, ϕ be an lcro-constraint, then $(l, \phi) \in Reach_C(A)$ iff $(l, \phi) \in Reach_{FP}(A)$.*

Proof. "$\Rightarrow$:" If (l, ϕ) is reachable in $[\![A]\!]_C$, then there exists a state (l, μ) such that (l, μ) is reachable in $[\![A]\!]_C$ and $\mu \models \phi$. Followed from the claim of Lemma 1, there exists a state $(l, (v, o))$ such that it is reachable in $[\![A]\!]_{FP}$, and $\mu \triangleright (v, o)$. Therefore, $v \models \phi$ by the Definition 6, then $(l, \phi) \in Reach_{FP}(A)$.

"$\Leftarrow$:" The proof is similar to the above using Lemma 2 and Definition 7. $\square$

In FPTAs, state equivalence is determined by the clock value and its order, which is different from the region equivalence in TAs [1]. The clock information of FPTAs $(v, o) \simeq (v', o')$, if they hold the following relation:

- for all $x \in X$, either $v(x) = v'(x)$, or $(v(x) \geq c_x + 1) \wedge (v'(x) \geq c_x + 1$ [2]$)$,
- for all $x \in X, o(x) = o'(x)$

The equivalence classification can ensure the finiteness of the state space. When the clock value of x is equal to or greater than $c_x + 1$, it is recorded as $c_x + 1$, which prevents the infiniteness of the state space. The models proposed in [22, 23] are similar to FPTAs, which use integer to record clock values and orders. However, they are based on the continuous semantics.

5 Reachability Analysis of FPTAs

One of the most common properties being checked by the verifiers is the reachability whose analysis is based on the exploration of the graph. There are two kinds of search strategies during state space exploration: forward and backward search. Currently our tool uses the forward search technique.

[2] c_x is the maximal constant in clock constraints on x in the automaton.

To relieve the state space explosion problem, verifiers usually use symbolic methods to record set of states. And the key issue is how to represent them. Different from the constraint based symbolic methods, the checker of FPTAs uses a data structure to describe the state set explicitly, meaning that all the sequences created by time delays are enumerated.

In this section, according to the characteristics of state space generation, we first analyze the features and the representation of a sequence of states created by time delays (called delay sequence). Then, based on the relation of the states generated from a segment of delay sequence by the discrete transition, we propose a data structure to represent the set of states symbolically. Thirdly, we present the symbolic transition systems of FPTAs. Finally, we introduce the algorithm for reachability analysis.

Let us fix for the rest of this section an FPTA $A = \langle L, l_0, \Sigma, X, I, E\rangle$.

5.1 Representation of States in the Delay Sequence

The generation of the state space is started from the initial state $(l_0, (v_0, o_0))$. Whenever allowed by the invariant of l_0, the sequence $(l_0, (v_0, o_0)) \rightarrow (l_0, (v_0, o_0) \oplus 1) \rightarrow \ldots$ can be generated by time delays, where $((v_0, o_0) \oplus i) \models I(l_0), i \geq 0$. Here we introduce a symbolic representation for this kind of sequence.

Definition 8. *(Symbolic Representation of Delay Sequence).*

- *Let $(l, (v, o), k)$ denote the set of states $\{(l, (v, o) \oplus 0), (l, (v, o) \oplus 1), \ldots, (l, (v, o) \oplus (k-1))\}$, where $k \in \mathbb{N}^{>0}$ ($\mathbb{N}^{>0}$ is the set of positive integers).*
- *Let $(l, (v, o), \infty)$ denote the set of states $\{(l, (v, o) \oplus 0), (l, (v, o) \oplus 1), \ldots\}$. Based on the equivalence relation, for all $x \in X$, all the clock valuations greater than $c_x + 1$ are treated as $c_x + 1$. Therefore, though time can progress infinitely, the number of states in the delay sequence is finite.*

We say that $(l, (v, o), k)$ is a delay sequence(DS) generated by delay transitions from the state $(l, (v, o))$, where $k \in \mathbb{N}^{>0} \cup \{\infty\}$.

In a delay sequence, when some states satisfy the guard of a transition, the corresponding discrete transition can be taken, leading to the new states. From these new states, the execution of delay or discrete transitions will be continued. So it is necessary to judge which state will satisfy the guard of the discrete transition. To compare every state with the guard is time-consuming. In this paper, we can get a set of such states rapidly according to the form of inequations in the guards.

According to the Definition 1, the guard is the conjunction of the inequations of the form $x \sim c, \sim \in \{<, \leq, >, \geq\}$. The inequations in the forms of $x > c$ and $x \geq c$ determine the minimum value the clock variable should be to switch to other locations; and inequations in the forms of $x < c$ and $x \leq c$ determine the maximal clock value to take discrete actions. By computing the sets of states satisfying the two kinds of constraints, we can get those that satisfy the guard of the discrete transition. Then the successors can be generated.

Next, we will generalize the features of the delay sequence. For example, the delay sequence formed from the initial state in Figure 1 is

$$(l_0, 0000, 5) = \{(l_0, 0000), (l_0, 1100), (l_0, 2200), (l_0, 3300), (l_0, 4400)\}.$$

With the increase of i, we can compute every state in the sequence respectively with $(v, o) \oplus i, 0 \leq i < 5$. Moreover, there is an ordering between the clock information of the states. Because of the convex nature of the constraints, if we find the state with the maximal clock value satisfying the constraints in the form of $x < c$ or $x \leq c$, all its pre-states in the delay sequence will meet the constraints too. Similarly, when we get the state with the minimum clock value satisfying the constraints in the form of $x > c$ or $x \geq c$, all its subsequent states in the delay sequence will satisfy this kind of constraints. The following definition describes how to determine the delay sequence restricted by constraints.

To facilitate the computation, let $c \in \mathbb{N}^+$, we assume that $\infty - c = \infty$, and $\infty > c$.

Definition 9. *(Constrained Delay Sequence). Let $(l, (v, o), k)$ be a delay sequence, and $\phi \in C(X)$ be the constraint, the constrained delay sequence $(l, (v, o), k)|_\phi$ is defined recursively as follows.*

1. *if ϕ is true, $(l, (v, o), k)|_\phi$ is $(l, (v, o), k)$.*
2. *if ϕ is $(x \leq c)$, let $l_c = \min\{(c - v(x)) * m + m - o(x), k\}$ where $m = 1 + \max o(X)$, then $(l, (v, o), k)|_{x \leq c}$ is $(l, (v, o), l_c)$.*
3. *if ϕ is $(x \geq c)$, let $d = (c - v(x)) * m - o(x)$, if $0 \leq d < k$, let $(v', o') = (v, o) \oplus d, g_c = k - d$, then $(l, (v, o), k)|_{x \geq c}$ is $(l, (v', o'), g_c)$. If $d \geq k$, then $(l, (v, o), k)|_{x \geq c}$ is empty. When $v(x) \geq c$, the original sequence keeps unchanged.*
4. *if $\phi = \phi_1 \wedge \phi_2$, $(l, (v, o), k)|_\phi$ is $((l, (v, o), k)|_{\phi_1})|_{\phi_2}$.*

Now let us consider the model in Figure 1 again, to interpret the function of the above computation in the state space generation process.

Example 2. Computation on the Constrained Delay Sequence.

Here, we try to determine the states capable of switching to l_1 by discrete transition t_1, with the guard $x < 5$ and $y \geq 2$. Since $I(l_0)$ is $y \leq 4$, the delay sequence started from the initial state is $(l_0, 0000, 5)$. First, the result of constrained delay sequence by $x < 5$ is $(l_0, 0000, 5)$; $(l_0, 2200, 3)$ is the result of $(l_0, 0000, 5)$ being constrained by $y \geq 2$, whose states can take discrete transition t_1. □

To check whether $(v_1, o_1) \in ((v, o), k)$, we can judge whether there exists k' such that $(v, o) \oplus k' = (v_1, o_1)$ and $k' < k$, where $k' = \max\{(v_1(x) - v(x)) * m + o_1(x) - o(x) | x \in X$ and $v_1(x) \leq c_x\}$. Let $(l, (v, o), k)$ and $(l, (v_1, o_1), k_1)$ be two sets of states, if $(v, o) \in ((v_1, o_1), k_1)$ or $(v_1, o_1) \in ((v, o), k)$, then the intersection of the two sets is not empty.

5.2 The Formation of Symbolic States

In the last subsection we have analyzed the features and the representation of the delay sequence. If we use it as the symbolic method to record the state space, the number of the symbolic states is still larger. However, if we can organize them into a coarser data structure, based on some relationship between these delay sequences, the number of the symbolic states can be reduced effectively.

In Figure 1, started from the initial state, the segment that can take discrete transition t_1 is $(l_0, 2200, 3) = \{(l_0, 2200), (l_0, 3300), (l_0, 4400)\}$. The occurrence of t_1 leads to the new set $\{(l_1, 2010), (l_1, 3010), (l_1, 4010)\}$. Among the new states, the clock information of y is $(0, 0)$ at every state, and the clock value of x increases monotonically.

Then given a state generated by the discrete transition, we can compute all other new successors from the states in the same delay sequence. Let (v_r, o_r) be the clock information after a discrete transition, where Y is the reset clock set. The clock information of the ith state from (v_r, o_r) is $(v_{ir}, o_{ir}) = ((v_r, o_r) \circledast i) \setminus Y$, where $(((v, o) \circledast i) \setminus Y)(x) =$

$$\begin{cases} (v(x) + (o(x) - 1 + i) \text{ div } m', (o(x) + i - 1) \text{ mod } m' + 1) & \text{if } x \notin Y \text{ and } Y \neq \emptyset \\ (0, 0) & \text{if } x \in Y \text{ and } Y \neq \emptyset \\ (v(x), o(x)) & \text{if } Y = \emptyset \end{cases},$$

and $m' = \max o(X)$.

Definition 10. *(Series of Delay Sequences). Let $(l, (v, o))$ be a state, $\theta \in \mathbb{N}^{>0}$ and $Y \subseteq X$, we use $(l, (v, o), \theta, Y)$ as the symbolic representation for the set of states $\{(l, (v', o')) | (v', o') = ((v, o) \circledast i) \setminus Y, 0 \leq i < \theta\}$. We call this representation the series of delay sequence (SDS). When $Y = \emptyset$, $(l, (v, o), \theta, Y)$ is $(l, (v, o), 1, \emptyset)$.*

For instance, $(l_1, 2010, 3, \{y\}) = \{(l_1, 2010), (l_1, 3010), (l_1, 4010)\}$ is a set of states in the example of Figure 1.

A symbolic state consists of a set of so-called start states, which are generated from states in the same delay sequence by a discrete transition. And new delay sequences will be generated from these start states.

In certain cases, some start states of the symbolic state may not satisfy the invariant of the new location. So we should determine the start states that meet the new invariant.

Definition 11. *(Constrained Symbolic State by Invariants $(l', (v, o), \theta, Y) \downharpoonleft_{I(l')}$). Let $(l', (v, o), \theta, Y)$ be a symbolic state, and $\phi = I(l')$ be the invariant of l'.*

- *if $Y = \emptyset$, assume that $(l', (v, o), \infty)|_\phi = (l', (v', o'), g)$, then $(l', (v, o), \theta, Y) \downharpoonleft_\phi$ is $(l', (v', o'), 1, \emptyset)$.*
- *if $Y \neq \emptyset$.*
  - *if ϕ is true, then $(l', (v, o), \theta, Y) \downharpoonleft_\phi$ is $(l', (v, o), \theta, Y)$.*
  - *if ϕ is $x \leq c$, and $x \notin Y$, let $\theta' = min\{\theta, (c - v(x)) * m' + m' - o(x) + 1\}$, then $(l', (v, o), \theta, Y) \downharpoonleft_{x \leq c}$ is $(l', (v, o), \theta', Y)$.*

- *if ϕ is $x \geq c$, and $x \notin Y$, let $d = (c - v(x)) * m' + 1 - o(x)$. If $0 \leq d < \theta'$, let $(v', o') = ((v, o) \circledast d) \setminus Y, \theta'' = \theta' - d$, then $(l', (v, o), \theta', Y) \downharpoonleft_{x \geq c}$ is $(l', (v', o'), \theta'', Y)$. If $d \geq \theta'$, $(l', (v, o), \theta', Y) \downharpoonleft_{x \geq c}$ is empty.*
- *if $\phi = \phi_1 \wedge \phi_2$, then $(l', (v, o), \theta, Y) \downharpoonleft_{\phi}$ is $((l', (v, o), \theta, Y) \downharpoonleft_{\phi_1}) \downharpoonleft_{\phi_2}$.*

The inclusion relation between the series of delay sequences can be judged quickly, which is similar to the judgement between delay sequences.

Let $(l, (v_1, o_1)), (l, (v_2, o_2))$ be two states which are generated from a delay sequence by the same discrete transition (with reset clock set $Y \neq \emptyset$), and $v_1(x) \leq v_2(x)$ for all $x \in X$. To simplify the representation, we introduce a function $len((v_1, o_1), (v_2, o_2), Y) =$

$$\begin{cases} \max\{(v_2(x) - v_1(x)) * m' + o_2(x) - o_1(x) | x \in Z\} & \text{if } Z \neq \emptyset \\ \max\{(c_x - v_1(x)) * m' + m' - o_1(x) | x \in X - Y\} & \text{if } Z = \emptyset \end{cases},$$

where $Z = \{x | v_2(x) \leq c_x, \text{and } x \in X - Y\}, m' = \max o_1(X)$.

Let $(l, (v, o), k)$ be a delay sequence meeting the guards of a discrete transition (l, ϕ, σ, Y, l'). Then the successor generated from $(l, (v, o), k)$ with transition (l, ϕ, σ, Y, l') is $(l', (v_r, o_r), \theta, Y)$, where

- $(v_r, o_r) = reset((v, o), Y)$,
- $\theta = dist((v, o), k, Y)$,
- $dist((v, o), k, Y) =$

$$\begin{cases} 1 & \text{if } Y = \emptyset \text{ or } Y = X \\ len((v_r, o_r), reset((v, o) \oplus (k-1), Y)) + 1 & \text{else if } k \neq \infty \\ \max\{(c_x - v_r(x) + 1) * m' - o_r(x) | x \in X - Y\} + 1 & \text{else} \end{cases},$$

- $m' = \max o_r(X)$.

For example, when the discrete transition t_1 in Figure 1 is taken, the delay sequence $(l_0, 2200, 3)$ will generate the new symbolic state $(l_1, 2010, 3, \{y\})$.

5.3 The Symbolic Transition Systems of FPTAs

The symbolic transition system of FPTA $A = \langle L, l_0, \Sigma, X, I, E \rangle$ is $\langle S, s_0, \Sigma, \rightsquigarrow \rangle$. $S = L \times \mathcal{D}$ is the set of symbolic states where $\mathcal{D} = \{((v, o), \theta, Y) | (v, o) \in (v_X \times o_X), Y \subseteq X, \theta \in \mathbb{N}^{>0}\}$. $s_0 = (l_0, D_0)$ is the initial symbolic state, where $l_0 \in L, D_0 = ((v_0, o_0), 1, \emptyset) \in \mathcal{D}$ is the initial symbolic clock information. The symbolic transition relation $\rightsquigarrow$ is defined as follows, which explains how the symbolic successor is created.

Definition 12. *(Symbolic Transition Relation $\rightsquigarrow$). Let $(l, D), (l', D') \in L \times \mathcal{D}$ be two symbolic states, where $D = ((v, o), \theta, Y), D' = ((v', o'), \theta', Y')$. We say $((l, D), (l', D')) \in \rightsquigarrow$, if there exist an integer $i \in \{0, 1, 2, \ldots, \theta - 1\}$ and a transition $e = (l, \phi, \alpha, Y', l') \in E$, such that $((v', o'), \theta', Y') = ((v_r, o_r), \theta_r, Y') \downharpoonleft_{I(l')}$, where*

- $(v_r, o_r) = reset((v'', o''), Y')$,

– $\theta_r = dist((v'', o''), g_\phi, Y')$, *and*
– $((v'', o''), g_\phi) = (((v, o) \circledast i) \setminus Y, \infty)|_{I(l) \wedge \phi}$.

With the state equivalent relation in FPTAs, the symbolic semantics results in a finite symbolic state space.

Example 3. The Generation of Symbolic States of the model in Figure 1.

The initial symbolic state is $s_0 = (l_0, 0000, 1, \emptyset)$. Then it just has one delay sequence $(l_0, 0000, 5)$, in which some states can take discrete transition t_1. The occurrence of t_1 leads to the symbolic state $s_1 = (l_1, 2010, 3, \{y\})$. Time can elapse in l_1 when its invariant is satisfied. Among the states resulting from time delays, only the state $(l_1, 5310)$ created from the start state $(l_1, 2010)$ can meet the guard of t_2. So the next symbolic state is $s_2 = (l_0, 0301, 1, \{x\}) \ldots$ □

5.4 Algorithms for Reachability Analysis

In this subsection we firstly present a reachability analysis algorithm. Then we list the generated symbolic states during the state space exploration in the example of Figure 1.

During the forward search of the state space, we use two lists W and P to record the states waiting for checking and being examined respectively. Another function of P is to avoid revisiting parts of the state space. W and P are empty in the beginning. Then the initial symbolic state is pushed into W. In every repetition, if the popped symbolic state from W has not been examined, the satisfiability of (l, ϕ) is checked. If it is satisfied, the whole process is completed. Otherwise, it is stored in P, and then successors are generated and pushed into W. The process can be described as Algorithm 1.

Algorithm 1 ReachabilityAnalysis

```
SDS: $wa$;
List of State: $Succ$;
List of SDS:$P, W := \emptyset$;
$wa := (l_0, (v_0, o_0), 1, \emptyset)$;
$W := \{wa\}$;
while $W \neq \emptyset$ do
  get $wa$ from $W$;
  if $wa \notin P$ then
    if Satisfy($wa, l, \phi$) then
      $return(true)$
    else
      Add($wa, P$);
      $Succ$ :=Unfolding($wa$);
      CreateSuccessive($Succ$)
    end if
  end if
end while
```

In Algorithm 1, *Satisfy* is *true* if there exists a state s in a delay sequence of a start state in wa such that $s \models (l, \phi)$. *Succ* stores the start states unfolded from the symbolic state by the *unfolding* function, which computes all the start states meeting the invariant of the current location. *CreateSuccessive* computes the successive symbolic states from the start states in *Succ*. If the new symbolic states have no inclusion relation with the states in P, they are pushed into W.

Here we use the example in Figure 1 to explain how the whole state space is generated. The steps to compute the symbolic states can refer to the definitions and examples above. We just list the symbolic states in P and W during the reachability analysis process in Table 1.

Table 1. Symbolic states in W and P

Step	W	P
1	$\{(l_0, 0000, 1, \emptyset)\}$	$\{\}$
2	$\{(l_1, 2010, 3, \{y\})\}$	$\{(l_0, 0000, 1, \emptyset)$
3	$\{(l_0, 0301, 1, \{x\})\}$	$\{(l_0, 0000, 1, \emptyset), (l_1, 2010, 3, \{y\})$
4	$\{(l_1, 0010, 2, \{y\})\}$	$\{(l_0, 0000, 1, \emptyset), (l_1, 2010, 3, \{y\}), (l_0, 0301, 1, \{x\})\}$
5	$\{(l_0, 0401, 1, \{x\})\}$	$\{(l_0, 0000, 1, \emptyset), (l_1, 2010, 3, \{y\}), (l_0, 0301, 2, \{x\}), (l_1, 0010, 2, \{y\})\}$
6	$\{\}$	$\{(l_0, 0000, 1, \emptyset), (l_1, 2010, 3, \{y\}), (l_0, 0301, 2, \{x\}), (l_1, 0010, 2, \{y\})\}$

In the 5th step of Table 1, the successor of $(l_1, 0010, 2, \{y\})$ is $(l_0, 0301, 2, \{x\})$. However, it includes the state $(l_0, 0301, 1, \{x\})$ in P. After P is updated, the unexamined part $(l_0, 0401, 1, \{x\})$ is pushed into W. The generated state in step 6 is already in P, so W is empty. Then the whole state space is generated.

6 Experiment

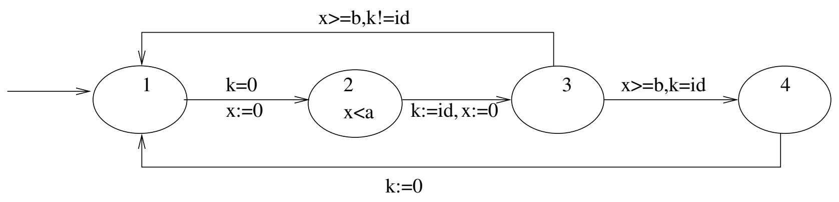

Fig. 2. Fischer's mutual exclusive protocol

We have implemented a prototype to support the verification of real-time systems with multi-processes, synchronizations, and broadcasts. The tool is available at http://lcs.ios.ac.cn/∼xyz/FPTA. Due to the page limit, we only use Fischer's mutual exclusive protocol [19] (see Figure 2) as the example. We compare the results with those of Uppaal 3.4.6 in various assignments of a and b, when the whole state space is created in breadth first search strategy. The environment is Intel P4 2.60GHz Dell PC with 512MB memory. In Table 2, we list the time consumption in seconds and the number of generated symbolic states (zones or SDSs) in the passed list. "-" indicates that the verification did not terminate within 600 seconds, and "N/A" stands for "not available".

Table 2. Results with Fischer's mutual exclusive protocol

no	Uppaal(a=2,b=4)		FPTA(a=2,b=4)		FPTA(a=9,b=19)	
	Time(s)	Zones	Time(s)	SDSs	Time(s)	SDSs
2	0.12	21	0	23	0.02	23
3	0.12	145	0.06	271	0.22	413
4	0.14	1073	0.61	1907	69.24	15185
5	1.13	8581	27.13	27155	-	N/A
6	41.09	75385	-	497980	-	N/A

The time and space consumption of Uppaal will not be quite different with various assignments of a and b. However, our prototype may be sensitive to the maximal constant of the constraints. And with the increasing number of processes or clock variables, the performance of FPTA checker is not so well as that of Uppaal, which has employed many methods to improve the efficiency of model checking [17]. Currently, except for the active clock reduction method [16], our prototype does not use other techniques to reduce the state space, or to compress the data yet. From this point of view, there is a great gap between the prototype and Uppaal. Though there are lots of work to be done, the discretization of FPTA is still a promising attempt, based on the preliminary results.

7 Related Work and Conclusion

Though FPTAs have integer clock valuations, they are different from the discrete time models, which do not concern about the ordering of events in a time unit. The model in [21] uses a global clock as the reference. The ordering of events happened in one time unit is distinguishable. But when the global clock reaches an integer point, all clock values will increase by 1. Then the ordering information disappears. FPTAs can keep the ordering until the next reset happens. The model in [14] uses integer clock valuations too. In its model, if no clock is reset in a discrete transition, all clock values will increase by 1. Otherwise, only reset clocks will be set to 0, and others will not change.

As to the data structures, recording time constraints in a DBM can reduce the sensitivity to the maximal constant of clock constraints. However, it can only express convex zones, and is not suitable for the data sharing. The BDD-like structures, like CDDs [4], REDs [22, 23], and CRDs [24], can express non-convex zones, and are easy to share the existed data. But the ordering between variables will affect the memory consumption greatly. An SDS enumerates all the delay sequences in the state space, which is different from the structures based on the constraints. Compared with DBMs, memory consumption is low in SDSs. And all operations on SDSs are very simple with linear complexity, lower than that of DBMs [7]. The shortcoming of SDSs is that the number of states described by one SDS may be smaller than that by a DBM. Compared with BDD-like data structures, SDSs avoid state space explosion caused by the inappropriate ordering between processes and variables. And they need not consider the normalization of different forms. But SDSs are sensitive to the number of clocks variables.

Based on the discussion, our future work will be as follows. Firstly, the performance of the prototype must be improved and more industrial examples should be carried out. Then the research on the features of FPTAs will be continued. Finally, the data structure needs to be adjusted to facilitate the data sharing, and to reduce the sensitivity to the number of clock variables and the maximal constant. SDSs are just an attempt as one of data structures of FPTAs, we will apply more mature techniques in discrete time models such as BDD data structure in the checker.

Acknowledgements. Thanks to Maciej Koutny for suggestions not only to the English but also to some technical content, and Yubo Xu and Chunming Liu for the implementation of the prototype. We also thank the anonymous referees for their helpful criticisms, comments, and suggestions.

References

1. Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science 126(2), (1994) 183-235
2. Alur, R., Henzinger, T.A.: A Really Temproal Logic. IEEE FOCS (1989) 164-169
3. Asarin, E., Bozga, M., Kerbrat, A., Maler, O., Pnueli, A., Rasse, A.: Data-Structures for the Verification of Timed Automata. In Proceedings of the International Workshop on Hybrid and Real-Time Systems, LNCS 1201 (1997) 346-360
4. Behrmann, G., Larsen, K.G., Weise, C., Wang, Y., Pearson, J.: Efficient Timed Reachability Analysis Using Clock Difference Diagrams. CAV (1999) 341-353
5. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
6. Bengtsson, J., Jonsson, B., Lilius, J., Wang, Y.: Partial Order Reductions for Timed System. CONCUR (1998) 485-500
7. Bengtsson, J., Wang,Y.: Timed Automata: Semantics, Algorithms and Tools. LNCS 3098 (2004) 87-124
8. Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P.: Systems and Software Verification: Model-Checking Techniques and Tools. Springer (2001)
9. Beyer, D., Lewerentz, C., Noack, A.: Rabbit: A Tool for BDD-based Verification of Real-Time Systems. CAV (2003) 122-125
10. Bosnacki, D., Dams, D., Holenderski, L.: A Heuristic for Symmetry Reductions with Scalarsets. LNCS 2021, FME (2001) 518-533
11. Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., Yovine, S.: Kronos: a Model-Checking Tool for Real-Time Systems. LNCS 1427, CAV (1998) 298-302
12. Bozga, M., Maler, O., Pnueli, A., Yovine, S.: Some Progress in the Symbolic Verification of Timed Automata. LNCS 1254, CAV (1997) 179-190

13. Bryant, R.: Graph-based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, 35(8) (1986) 677-691
14. Dang, Z., Ibarra, O.H., Bultan, T., Kemmerer, R.A., Su, J.: Binary Reachability Analysis of Discrete Pushdown Timed Automata. LNCS 1855, CAV (2000) 69-84
15. Daws, C., Olivero, A., Tripakis, S., Yovine, S.: The tool KRONOS. Hybrid Systems III, LNCS 1066 (1996) 208-219
16. Daws, C., Yovine, S.: Reducing the Number of Clock Variables of Timed Automata. IEEE RTSS (1996) 73-81
17. Gerd, B., Johan, B., Alexandre, D., Larsen, K.G., Paul, P., Wang, Y.: UPPAAL Implementation Secrets. FTRTFT (2002) 3-22
18. Hendriks, M., Behrmann, G., Larsen, K.G., Vaandrager, F.: Adding Symmetry Reduction to Uppaal. FORMATS (2003) 46-59
19. Lamport, L.: A Fast Mutual Exclusion Algorithm. ACM Transactions on Computer Systems, 5(1), (1987) 1-11
20. Larsen, K.G., Pettersson, P., Wang, Y.: UPPAAL in a Nutshell. International Journal on Software Tools for Technology Transfer, 1(1/2), (1997) 134-152
21. Raskin, J.F., Schoebbens, P.: Real-Time Logics: Fictitious Clock as an Abstraction of Dense Time. LNCS 1217, TACAS (1997) 165-182
22. Wang, F.: Efficient Data Structure for Fully Symbolic Verification of Real-Time Software Systems. TACAS (2000) 157-171
23. Wang, F.: Region Encoding Diagram for Fully Symbolic Verification of Real-Time Systems. COMPSAC (2000) 509-515
24. Wang, F: Efficient Verification of Timed Automata with BDD-like Data-Structures. VMCAI (2003) 189-205
25. Wang, F.: Formal Verification of Timed Systems: A Survery and Perspective. In Proceedings of the IEEE, 92(8), (2004) 1283-1307

A Proofs of Lemma 1 and 2

A.1 Lemma 1

Proof. According to the definition of transition relations, it suffices to show the following:

- if $\mu \triangleright (v, o)$, then for all $d \in \mathbb{R}^+$, there exists $k \in \mathbb{N}^+$ such that $(\mu + d) \triangleright (v, o) \oplus k$.
- if $\mu_1 \triangleright (v_1, o_1)$ and $(l_1, \mu_1) \xrightarrow{\sigma} (l_2, \mu_2)$, then there exists (v_2, o_2) such that $\mu_2 \triangleright (v_2, o_2)$ and $(l_1, (v_1, o_1)) \xrightarrow{\sigma} (l_2, (v_2, o_2))$.

Firstly, we prove that the states resulting from delay transitions in FPTAs and TAs satisfy the $\triangleright$ relation. Suppose this is not true, and $\mathcal{D}$ is the set of all d such that there is no $k \in \mathbb{N}^+$ such that $(\mu + d) \triangleright (v, o) \oplus k$. Let d_0 be the infimum of $\mathcal{D}$, then $d_0 \in \mathcal{D}$.

Let $Z = \{x \in X \mid \mathrm{frac}(\mu(x) + d_0) = 0\}$, then Z is the set of clocks whose values will become integer after x increased by d_0. If Z is empty, no clocks will be integer when time increased by d_0. Then $(\mu + d_0) \triangleright (v, o)$, contrary to the hypothesis. So Z is nonempty.

Since clock values increased by d_0 will cause the change of fractional parts of clock values, so will the ordering. Let $\delta_2 = \frac{1}{2}\min\{\{1\} \cup \{\text{frac}(\mu(x) + d_0) \mid x \in X - Z\}\}$, then $(\mu + d_0 - \delta_2)$ will not affect the ordering of the fractional parts.

If there exists k_1, such that $(\mu + d_0 - \delta_2) \rhd (v, o) \oplus k_1$, then let $k_2 = 1 + \max\{o_1(x) - o_1(y) \mid x, y \in Z\}$, where $o_1 = o \oplus k_1$. Therefore, $(\mu + d_0) \rhd (v, o) \oplus (k_1 + k_2)$, contrary to the fact that $d_0 \in \mathcal{D}$.

It is straightforward to prove the states generated by discrete transitions hold the relation. Let $(v_2, o_2) =$reset$((v_1, o_1), Y)$, then $\mu_2 \rhd (v_2, o_2)$, where Y is the reset clock set in the transition. □

A.2 Lemma 2

Proof. According to the definition of transition relations, it suffices to show the following:

- if $(v, o) \simeq \mu$, then for all $k \in \mathbb{N}^+$, there exists $d \in \mathbb{R}^+$ such that $(v, o) \oplus k \simeq (\mu + d)$.
- if $(v_1, o_1) \simeq \mu_1$ and $(l_1, (v_1, o_1)) \xrightarrow{\sigma} (l_2, (v_2, o_2))$, then there exists $d \in \mathbb{R}^+$ such that $(v_1, o_1) \simeq (\mu_1 + d)$, $(l_1, \mu_1 + d) \xrightarrow{\sigma} (l_2, \mu_2)$ and $(v_2, o_2) \simeq \mu_2$.

Firstly we prove that the states resulting from delay transitions in FPTAs and TAs satisfy the $\simeq$ relation. Assume this is not true, and $\mathcal{K}$ is the set of all k such that there is no $d \in \mathbb{R}^+$ such that $(v, o) \oplus k \simeq (\mu + d)$. Let k_0 be the least element in $\mathcal{K}$, then $k_0 > 0$ (otherwise $(v, o) \oplus 0 \simeq \mu$, then $k_0 \notin \mathcal{K}$, contrary to the fact that $k_0 \in \mathcal{K}$).

If there exists d_1, such that $(v, o) \oplus (k_0 - 1) \simeq (\mu + d_1)$, then let $d_2 = \min\{1 - \text{frac}(\mu(x) + d_1) \mid x \in X\}$. So after time increased by $d_1 + d_2$, the clock value with the largest fractional part will become integer. Therefore,$(v, o) \oplus (k_0 - 1 + 1) = (v, o) \oplus k_0 \simeq (\mu + d_1 + d_2)$, contrary to the fact that $k_0 \in \mathcal{K}$.

Now we prove the second claim. Let $d = \frac{1}{2}\min\{1 - \text{frac}(\mu(x)) | x \in X\}$. Since $\min\{1 - \text{frac}(\mu(x))\}$ is the real number that causes some clock values to become integer, $\mu + d$ will not affect the relative ordering of fractional parts of clock values. So $(v_1, o_1) \simeq (\mu_1 + d)$.

Let $\mu_2 = (\mu_1 + d)[Y := 0]$ where Y is the clock set to be reset, then $(l_1, \mu_1 + d) \xrightarrow{\sigma} (l_2, \mu_2)$ and $(v_2, o_2) \simeq \mu_2$. □