SYMBOLIC VERIFICATION OF COMPLEX REAL-TIME SYSTEMS WITH CLOCK-RESTRICTION DIAGRAM

Farn Wang
Institute of Information Science, Academia Sinica
Taipei, Taiwan 115, ROC
farn@iis.sinica.edu.tw

Abstract Real-world real-time systems may involve many complex structures, which are difficult to verify. We experiment with the model-checking of an application-layer html-based web-camera which involves structures like event queues, high-layer communication channels, and time-outs. To contain the complexity, we implement our verification tool with a newly developed BDD-like data-structure, reduced CRD (Clock-Restriction Diagram), which has enhanced the verification performance through intensive data-sharing in a previous report. However, the representation of reduced CRD does not allow for quick test of zone containment. To this purpose, we thus have designed a new CRD-based representation, cascade CRD, which has given us enough performance enhancement to successfully verifying several implementations of the web-camera.

Keywords: Real-time systems, BDD, verification, Clock-Restriction Diagram

1. INTRODUCTION

Fully symbolic model-checking with BDD data-structure[8, 4] has achieved great success in hardware systems and become one of the standard technologies in Electronic Design Automation (EDA). Naturally, people are now looking forward to repeating the success in model-checking complex systems, which may involve structures like queues, stacks, clocks, communicaiton channels, ... Such structures usually cause great complexities to verify with BDD-like data-structures. In this work, we experiment with a real-world project, an html-based web-camera (`http://www.mmedia.com.tw`) implemented in application layer, and discuss how we have coped with the challenges in verification. Especially, we implemented the newly developed CRD (Clock-Restriction Diagram), which is a BDD-like data-structure for space-efficiency in timed automaton verification, in our model-checking tool and found that the representation

of reduced CRD does not allows for efficient testing of zone containment. It was only with our new representation scheme of cascade CRD that we had been able to verify implementations of the web-cameras.

Most modern model-checkers for real-time systems are built around some symbolic manipulation procedures[12] of zones implemented in data-structures like DBM[10], NDD[1], CDD[7], RED[15], or CRD[16]. A zone means a behaviorally equivalent state subspace of a timed automaton and is symbolically represented by a set of difference constraints between clock pairs. It has been the general opinion that DBM (difference-bounded matrix) [10] is perhaps the best data-structure for the representation and manipulation of zones. For a long time, BDD-like structures[1, 7, 15, 16] have not performed as well as the popular DBM (difference-bounded matrix) [10] which is a 2-dimensional matrix and nothing BDD-like. But recently, there comes a new BDD-like data-structure, called CRD (Clock-Restriction Diagram) [17], which has shown efficiency in experiments (see appendix 6). CRD uses evaluation variables corresponding to the entries in DBM. Particularly, the default value of variables, say $x - x'$, is treated as $x - x' < \infty$ (i.e. no restriction). Since a CRD is actually a DBM set represented in decision diagram, it is neither a canonical representation of dense-domain state-spaces. The representation of CRD with zones with smallest set of inequalities[14], i.e. reduced CRD, was proposed in [17] to cope with space-complexity. But the drawback of reduced CRD is that zone containment cannot be determined efficiently. Thus many many zones, which are contained by others in the reachable state space representation, are calculated again and again. As a result, both space and time-complexities become unmanageable and our verification tool with reduced CRD could not finish verifying several implementations of the web-camera.

Note that the problem, with the zone containement relation, mentioned in the last paragraph cannot be simply handled by using the representation of CRD with all-pair shortest-path representation zones [10, 14] because as argued in [17], such representation is less space-efficient than reduced CRD. To overcome this obstacle, we here propose a new representation of CRD, the cascade CRD, which is not a representation of set of zones with minimal number of inequalities. For a zone with identical clock readings, cascade CRD may contain more inequalities than reduced CRD. But for a given state-space (i.e. a set of zones), cascade CRD may result in much less zones recorded than reduced CRD with efficient elimination of zones contained by others. It is with the cascade CRD that we have successfully verified several implementations of the web-camera system.

Here is our presentation plan. Section 2 briefly defines timed automata as our model for discussion. Section 3 basically restates part of [17] to give readers a background knowledge to the issues in verification research of timed automata and CRD. Section 4 presents cascade CRD and its computation. Section 5

reports our tool implementations and experiments with the web-camera. Appendix 6 shows how good our CRD-based verification tool is compared to two well-known verification tools and justifies our choice to continue experiments with CRD.

2. TIMED AUTOMATA

We use the widely accepted timed automata[2] as our model. A timed automaton is a finite-state automaton equipped with a finite set of clocks which can hold nonnegative real-values. It is structured as a directed graph whose nodes are modes (control locations) and whose arcs are transitions. The modes are labeled with invariance conditions while the transitions are labeled with triggering conditions and a set of clocks to be reset during the transitions. The invariance conditions and triggering conditions are Boolean combinations of inequalities comparing a clock with an integer. At any moment, the timed automaton can stay in only one mode. In its operation, one of the transitions can be triggered when the corresponding triggering condition is satisfied. Upon being triggered, the automaton instantaneously transits from one mode to another and resets clocks in the corresponding transition clock set label to zero. In between transitions, all clocks increase their readings at a uniform rate.

For convenience, given a set X of clocks, we use $B(X)$ as the set of all Boolean combinations of inequalities of the form $x - x' \sim c$ where $x, x' \in X \cup \{0\}$, "$\sim$" is one of $\leq, <, =, >, \geq$, and c is an integer constant.

Definition 1 automata A timed automaton A is given as a tuple $\langle X, Q, q_0, I, \mu, T, \gamma, \tau, \pi \rangle$ with the following restrictions. X is the set of clocks. Q is the set of modes. q_0 is the initial mode. $I \in B(X)$ is the initial condition on clocks. μ: $Q \mapsto B(X)$ defines the invariance condition of each mode. T is a finite set of transitions. $\gamma : T \mapsto (Q \times Q)$ describes the source and destination modes of transitions. $\tau : T \mapsto B(X)$ and $\pi : T \mapsto 2^X$ respectively defines the triggering condition and the clock set to reset of each transition. ||

A valuation of a set is a mapping from the set to another set. Given an $\eta \in B(X)$ and a valuation v of X, we say v satisfies η, in symbols $v \models \eta$, iff when the variables in η is interpreted according to v, η will be evaluated true.

Definition 2 states Given a timed automaton $A = \langle X, Q, q_0, I, \mu, T, \gamma, \tau, \pi \rangle$, A state v of A is a valuation of $X \cup \{\texttt{mode}\}$ such that

- $v(\texttt{mode}) \in Q$ is the mode of A in v with mode as a special auxiliary variable; and
- for each $x \in X$, $v(x) \in R^+$ such that R^+ is the set of nonnegative real numbers and $v \models \mu(v(\texttt{mode}))$. ||

For any $t \in R^+$, $v+t$ is a state identical to v except that for every clock $x \in X$, $v(x) + t = (v+t)(x)$. Given $\bar{X} \subseteq X$, $v^{\bar{X}}$ is a new state identical to v except that for every $x \in \bar{X}$, $v^{\bar{X}}(x) = 0$.

Definition 3 runs Given a timed automaton $A = \langle X, Q, q_0, I, \mu, T, \gamma, \tau, \pi \rangle$, a v-run is an infinite sequence of state-time pair $(v_0, t_0)(v_1, t_1) \ldots (v_k, t_k) \ldots\ldots$ such that $v = v_0$ and $t_0 t_1 \ldots t_k \ldots\ldots$ is a monotonically increasing real-number (time) divergent sequence, and for all $k \geq 0$,

- for all $t \in [0, t_{k+1} - t_k]$, $\nu_k + t \models \mu(\nu_k\ (\texttt{mode}))$; and
- either $v_k\ (\texttt{mode}) = v_{k+1}\ (\texttt{mode})$ and $v_k + (t_{k+1} - t_k) = v_{k+1}$; or for some $w \in T$,
  - $\gamma(w) = (\nu_k(\textbf{mode}), \nu_{k+1}(\textbf{mode}))$; and
  - $\nu_k + (t_{k+1} - t_k) \models \tau(w)$; and
  - $(\nu_k + (t_{k+1} - t_k))\pi(w) = \nu_{k+1}$. ||

A safety requirement on timed automaton A can be written as a Boolean combination of clock constraints in $B(X)$ and mode restrictions in the form of $\texttt{mode} = q$ meaning that A is currently in mode $q \in Q$. A run $\rho = (\nu_0, t_0)(\nu_1, t_1) \ldots (\nu_k, t_k)$ ..... of A satisfies safety requirement η, in symbols $\rho \models \eta$, iff for all $k \geq 0$ and $t_k \leq t \leq t_{k+1}$, $\nu_k + t \models \eta$. We say $A \models \eta$ iff for all v-runs ρ, $v \models I \wedge (\texttt{mode} = q_0)$ implies $\rho \models \eta$. Our verification framework is safety analysis problem that when given A and η, asks whether $A \models \eta$.

3. BASICS

To better prepare readers for understanding of the materials, we restate some of the paragraphs form [17] in this section. Subsection 3.1 discusses the concept of zones. Subsection 3.2 formally defines CRD and its manipulations. Especially, subsection 3.2.2 illustrates how reduced CRD can be more space-efficient than the other technologies.

3.1 Zones, closure form, and reduced form

Most modern model-checkers are built around some symbolic manipulation procedures[12] of zones implemented in data-structures like DBM, NDD, CDD, RED, or CRD, A zone means a behaviorally equivalent state subspace of a timed automaton and is symbolically represented by a set of difference constraints between clock pairs. Previously, people still believed that DBM was the most efficient data-structure. DBM-technology generally handles the complexity of timing constant magnitude very well. But when the number of clocks increases, its performance also degrades rapidly.

Let $\mathcal{Z}$ be the set of integers. Given $c \geq 0$ and $c \in \mathcal{Z}$, let I_c be $\{\infty\} \cup \{ d \mid d \in \mathcal{Z}; -c \leq d \leq c \}$. Also for any $d \in \mathcal{Z}$, $d + \infty = \infty + d = \infty$.

Given a safety analysis problem for a timed automaton A with biggest timing constant C_A used in A, a zone is a convex subspace of $R^{|X|}$ constrained by half spaces represented by inequalities like $x - x' \sim d$, with $x, x' \in X \cup \{0\}$, $\sim \in \{\text{“}\leq\text{”}, \text{“}<\text{”}\}$, and $d \in I_{C_A}$, such that when $d = \infty$, $\sim$ must be “$<$”. For convenience, let $\mathcal{B}_c = \{(\sim, d) \mid \sim \in \{\text{“}\leq\text{”}, \text{“}<\text{”}\}; d \in \mathcal{I}_c; d = \infty \Rightarrow \sim = \text{“}<\text{”}\}$. With respect to given X and C_A, the set of all zones is finite. Formally, a zone ς can be defined as a mapping $(X \cup \{0\})^2 \mapsto B_{C_A}$. Alternatively, we may also define a zone ς as the set $\{x - x' \sim d \mid \varsigma(x, x') = (\sim, d)\}$. In the following, we shall use the two equivalent definitions flexibly as we see fit.

There can be many zones representing the same convex subspace. A straightforward canonical representation of a zone-characterizable convex subspace is its zone in closure form (called shortest-path closure in [14]). A zone ς is in closure form if and only if for any sequence of elements $x_1, \ldots, x_k \in X \cup \{0\}$, with $x_1 - x_k \sim d \in \varsigma$ and $\forall 1 \leq i < k (x_i - x_{i+1} \sim_i d_i \in \varsigma)$, either $d < \sum_{1 \leq i < k} d_i$ or $(d = \sum_{1 \leq i < k} d_i \wedge (\sim = \text{“}\leq\text{”} \Rightarrow \bigwedge_{1 \leq i < k} \sim_i = \text{“}\leq\text{”}))$. Intuitively, this means that every half space constraint has to be tight. We can artificially designate the closure form of each zone as our canonical representation of the corresponding state subspace characterized by the zone. For convenience, given a zone ς, we let ς^C be the notation for its closure form.

A few terms to define before we explain the second candidate for zone canonical representation. Two clocks $x, x' \in X \cup \{0\}$ are equivalent in a zone ς, in symbols $x \equiv_\varsigma x'$, iff $\exists d \in \mathcal{Z}(x - x' \leq -d \in \varsigma^C \wedge x' - x \leq d \in \varsigma^C)$. For convenience, assume that $X = \{x_1, \ldots, x_n\}$ and 0 is also named x_0. If $Y = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\} \subseteq X \cup \{0\}$ is a maximal set of equivalent clocks in ς such that $i_1 < i_2 < \ldots < i_k$, let $\min_{\equiv\varsigma}(x_{i_j}) = x_{i_1}$ for all $1 \leq j \leq k$. Given $x - x' \sim d \in \varsigma^C$, (x, x') is redundant in ς iff $x \not\equiv_\varsigma x'$ and there is a $\bar{x} \in X \cup \{0\}$, with $x - \bar{x} \sim_1 d_1, \bar{x} - x' \sim_2 d_2 \in \varsigma^C$, such that $x \not\equiv_\varsigma \bar{x} \not\equiv_\varsigma x'$ and $d = d_1 + d_2 \wedge (\sim = \text{“}<\text{”} \Rightarrow (\sim_1 = \text{“}<\text{”} \vee \sim_2 = \text{“}<\text{”}))$.

Another candidate for the canonical representation of zones is the reduced form (called shortest-path reduction in [14]) which records only minimum number of constraints for each zone. We refer interested readers to [14, 17] for explanation how to convert a given zone V to its zone in reduced form, in symbols ς^R. It is shown in [14] that $\varsigma^C = (\varsigma^R)^C$; and DBM with zones in reduced form can be used as a canonical representation of timed automaton convex states-spaces and can significantly save space in model-checking.

3.2 Clock Restriction Diagram

CRD is not a decision diagram for state space membership. Instead it is like a database for zones. We devise the new data-structure CRD exactly because CRD acts like a database (recording device) and is more suitable for comparison and manipulation of sets of clock diference constraints.

3.2.1 Previous data-structures. NDD[1] uses binary encoding for clock readings and its performance is very sensitive to timing-constant magnitude.

CDD[7] is a decision diagram for state-space membership. It has very similar structure to CRD. There are two major differences of CRD. First, a value d for evaluation variable, say $x - x'$, in CDD means $x - x' = d$ while a value like $(\sim, d)$ means $x - x' \sim d$. This is because CDD is designed to be decision diagram. Second, the default value of a variable in CRD is interpreted as $(<, \infty)$ while it is interpreted as $(-\infty, \infty)$ in CDD. To reduce information duplication in representations of state-spaces, CDD has to be transformed to closure form [10, 7] which records the shortest-path distances between all pairs of clocks and is very space-inefficient.

RED[15, 16] encodes the ordering of fractional parts of clock readings in the variable ordering and has achieved very high space-efficiency for systems with large number of clocks and small timing constants. RED is indeed a canonical representation of timed automaton state subspaces. But for large timing constants, RED's performance degrades rapidly.

At this moment, DBM is still the most popular and efficient data-structure. DBM-technology generally handles the complexity of timing constant magnitude very well. But when the number of clocks increases, its performance also degrades rapidly.

3.2.2 Definition. CRD is a directed acyclic graph for representation of sets of zones. It has similar structure as BDD without FALSE terminal. Each of the pairs $(x, x') \in (X \cup \{0\})^2$ is treated as an evaluation variable. By fixing an evaluation order, we can construct a CRD just as BDD, CDD, or RED. For example, given $C_A = 10$, the CRD for a set $\{\{0 - x_1 \leq -3, x_1 - x_3 < -4\}, (0 - x_2 < -1, x_2 - x_1 < 6\}\}$ of two zones (constraints of the form $x - x' < \infty$ are omitted) is in figure 1(a). In CRD, a missing constraints on differences of clock pairs, say x,x', is interpreted as $x - x' < \infty$. Thus in the root vertex, even no constraint is on $0 - x_1$ in the latter zone, we still construct an arc with $0 - x_1 < \infty$ from the root vertex. This is one major difference of our CRD from decision diagrams like CDD which interprets a missing restriction on x,x' as $-\infty < x - x' < \infty$ with an implied lowerbound of $-\infty$ on $x - x'$.

An evaluation index $\omega : (X \cup \{0\})^2 \cup \{\text{true}\} \mapsto \{0, 1, \ldots, |(X \cup \{0\})^2|\}$ is a mapping such that $\omega(\text{true}) = |(X \cup \{0\})^2|$ and for every two $e, e' \in (X \cup \{0\})^2 \cup \{\text{true}\}, \omega(e) \neq \omega(e')$.

Definition 4 Clock Restriction Diagram (CRD) A CRD is a labeled directed acyclic graph $D = (V, \phi, E, \lambda)$, with single source and single sink, constructed under a given evaluation index w such that

- V is the set of vertices;
- $\phi : V \mapsto (X \cup \{0\})^2 \cup \{\text{true}\}$ defines the evaluation variable at each vertex;

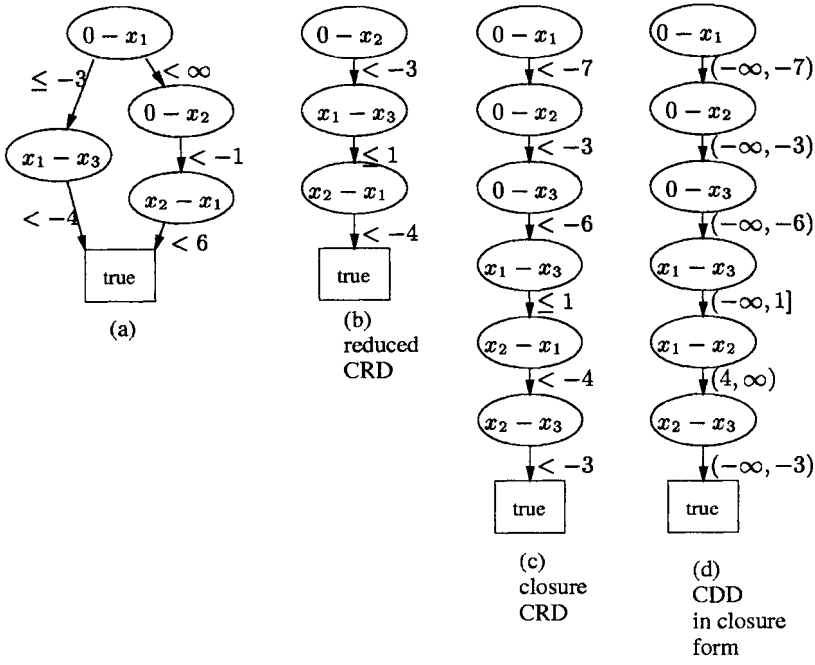

Figure 1. examples of CRD and comparison with CDD

- $E \subseteq V \times V$ is the set of arcs such that for every $(v, v') \in E, \omega(\phi(v)) < \omega(\phi(v')$(i.e., evaluation ordering must be respected); and
- $\lambda : E \mapsto \mathcal{I}_{C_A}$ such that for every $(v, v'), (v, v'') \in E$, $v' \neq v'' \Rightarrow \lambda(v, v') \neq \lambda(v, v'')$

There is at most one $v \in V$ such that $\phi(v)$ = true and this v is the single sink of the CRD. ‖

Since many zones can represent the same subspace, like DBM and CDD, neither is CRD a canonical representation of zone-characterized state spaces. To reduce information duplication in state-space representation, one solution is to convert all zones to their closure form[10,7] (called shortest-path closure in [14]), which is the set of all pairwise clock difference constraints derived from the all-pair shortest-path distances, and only store their closure form. Such conversion is expensive and, as we shall illustrate in figure 1(b)-(d), incurs large space consumption with data-structures like CDD and CRD.

Also note that although zones in closure form can be used as canonical representation for convex state-spaces characterizable by zones, CRD with closure form zones is not a canonical representation of dense-time state-spaces. This is

because CRD with closure form zones does not intrinsically have the capability to eliminate zones contained by the unions of other zones.

An alternative solution for space-efficient representation of state spaces is zones in their reduced form (called shortest-path reduction in [14]) which contains minimal number of clock difference constraints chosen by a policy. As shown in [14], DBM with zones in reduced form can be space-efficient. Wang has proposed to use CRD with zones in reduced form (or reduced CRD in short) as the representation for state-spaces to enhance verification efficiency [17]. Moreover, a space-efficient algorithm, with only four auxiliary variables, for computing reduced CRD has also been presented in [17].

Note again that reduced CRD is neither a canonical representation of dense-time state-space. Nevertheless, in average, CRD with zones in reduced form (reduced CRD in short) is much more space-efficient than previous data-structures [1, 7, 15, 16]. For example, given $X = \{x_1, x_2, x_3\}$ and $C_A = 10$, in figure 1(b),(c),(d), we have the representations of zone $\{0 - x_2 < -3, x_1 - x_3 \leq 1, x_2 - x1 < -4\}$ in reduced CRD (figure 1 (b)), in CRD with zones in closure form (figure 1(c)), and in CDD with zones in closure form (called tightened form in [7]) without FALSE terminal vertex (figure 1(d)). It is easy to see that as the number of clocks increases, reduced CRD will perform better and better.

3.2.3 Set-oriented manipulations on CRD. For convenience of discussion, given a CRD, we may just represent it as the set of zones recorded in it. Set-union ($\cup$), set-intersection ($\cap$), and set-exclusion (–) of two zone sets respectively represented by two CRDs are straightforward. For example, given CRDs $D_1 : \{\zeta_1, \zeta_2\}$ and $D_2 : \{\zeta_2, \zeta_3\}$, $D_1 \cap D_2$ is the CRD for $\{\zeta_2\}$; $D_1 \cup D_2$ is the CRD for $\{\zeta_1, \zeta_2, \zeta_3\}$; and $D_1 - D_2$ is the CRD for $\{\zeta_1\}$. The complexities of the three manipulations are all $O(|D_1| \cdot |D_2|)$.

Set-extraction ($|$) selects zones satisfying certain features from a zone set. Suppose D is the CRD for $\{\zeta_1, \ldots, \zeta_k\}$ and D' is the CRD for $\{\zeta'_1, \ldots, \zeta'_h\}$, then $D|D' = \{\zeta_i \mid 1 \leq i \leq k; \exists 1 \leq j \leq h \forall x - x' \sim d \in \zeta'_j (d \neq \infty \Rightarrow x - x' \sim d \in \zeta_i)\}$ The complexity is also $O(|D_1| \cdot |D_2|)$.

Given two zones ζ_1 and ζ_2, $\zeta_1 * \zeta_2$ is a new zone representing the space-intersection of ζ_1 and ζ_2. Formally speaking, for every x, x' with $\zeta_1(x, x') = (\sim_1, d_1)$ and $\zeta_2(x, x') = (\sim_2, d_2)$, $\zeta_1 * \zeta_2(x, x') = (\sim_1, d_1)$ if $d_1 < d_2 \vee (d_1 = d_2 \wedge \sim_1 = \text{“<”})$; or $\zeta_1 * \zeta_2(x, x') = (\sim_2, d_2)$ otherwise. Space-intersection (*) of two CRDs D_1 and D_2, in symbols $D_1 * D_2$, is a new CRD for $\{\zeta_1 * \zeta_2 \mid \zeta_1 \in D_1; \zeta_2 \in D_2\}$. Our current implementation of the manipulation has complexity $O(|D_1|^2 \cdot |D_2|^2)$.

3.2.4 CRD and BDD. It is possible to combine CRD and BDD into one data-structure for fully symbolic manipulation. Since CRD only has one sink vertex: true, it is more compatible with BDD without FALSE terminal vertex

which is more space-efficient than ordinary BDD. There are two things we need to take care of in this combination. The first is about the interpretation of default values of variables. In BDD, when we find a variable is missing during valuating variables along a path, the variable's value can be interpreted as either TRUE or FALSE. But in CRD, when we find a variable for constraint $x - x'$ is missing along a path, then the constraint is interpreted as $x - x' < \infty$.

The second is about the interpretation of CRD manipulations to BDD variables. Straightforwardly, "$\cup$" on Boolean variables is interpreted as "$\vee$" on Boolean variables. "$\cap$" and "$|$" on Boolean variables are both interpreted as "$\wedge$" on Boolean variables. $D_1 - D_2$ on Boolean variables is interpreted as $D_1 \wedge \neg D_2$ when the root variable of either D_1 or D_2 is Boolean. For $D_1 * D_2$, the manipulation acts as "$\wedge$" when either of the root variables are Boolean. Due to page-limit, we shall omit the proof for the soundness of such interpretation.

From now on, we shall call it CRD+BDD a combination structure of CRD and BDD.

3.2.5 Variable ordering in CRD. In our BDD+CRD, we found the following evaluation ordering quite efficient in our experiment.

- All discrete variables precede those clock inequality variables in the evaluation ordering.
- Let $0 \prec x_1 \prec \ldots \prec x_n$. For clocks $x, x', y, y' \in \{0, x_1, \ldots, x_n\}$, $x - x'$ precedes $y - y'$ iff either (1) $(x \prec y \wedge x \prec y') \vee (x' \prec y \wedge x' \prec y')$; or (2) $x \prec y \wedge x = y' \wedge x' = y$.

Especially, the condition (2) of item 2 puts variable like $x_2 - x_1$ immediately below $x_1 - x_2$ and allows us to efficiently check for some trivial negative cycles.

4. CASCADE CRD

Reduced CRD indeed can be very space-efficient, as reported in [17], for some applications. But it is more difficult to efficiently decide the containment relation between two zones with reduced CRD. For example, we may have the reduced CRD in figure 2(a) for the state-space represented by formulus (1) in the following.

$$\begin{array}{ll} & (x_1 - x_2 \leq -2 \wedge x_2 - x_4 \leq -3 \wedge x_4 - x_1 \leq 5 \wedge x_1 = x_3) \\ \vee & (x_1 - x_2 \leq -2 \wedge x_2 - x_4 \leq -3 \wedge x_4 - x_1 \leq 5) \end{array} \quad (1)$$

As can be seen that the zone of the first conjunction is actually contained in that of the second. This containment relation can be easily computed with all-pair shortest-path difference relation between clock pairs. But when we transform a given CRD into its reduced CRD, there is not enough information left to efficiently derive such containment relation. In specific, we cannot use the simple reduce operations described in the last paragraph of subsection 3.2.3 to eliminate zones which are contained by others in the same CRD. Thus, many

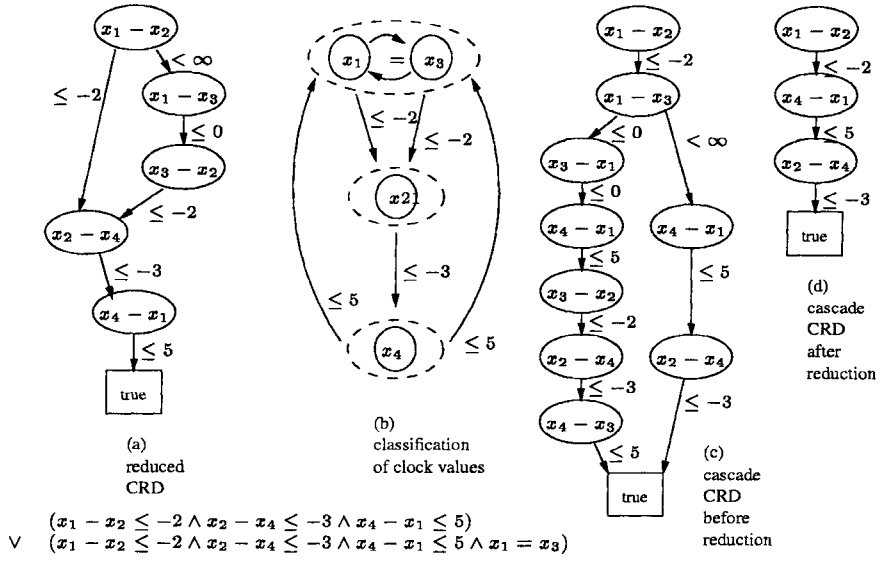

Figure 2. reduction in cascade CRD

many zones may be represented in CRD for the reachable zone set which are actually subsets of zones already generated before. This problem may result in huge waste in both space and computation time. Actually, in our experiment, none of the web-camera implementations can be verified with reduced CRD.

Here we propose another representation scheme of CRD, called cascade CRD, which allows us to efficiently control CRD complexity with the simple reduction operations mentioned at the end of subsection 3.2.3. For the convenience of defining cascade CRD, we first define the cascade form of zones. A cascade CRD is merely a CRD, all whose zones are represented in their cascade form. Just like closure form and reduced form of zones, the cascade form of a zone is also a canonical representation of zone-characterizable state-spaces. The idea is to add a few more inequalities to zones in reduced forms to facilitate the decision of zone containment relation. Cascade CRD is especially designed for those systems which can have states in which many clocks have identical values. This kind of states may happen because of clock-reset operations at a synchronization, between a sender and a receiver, which is very common in modelling tightly-coupled interactions in concurrent systems.

In figure 2(b), we illustrate how we compute the cascade zone from its non-cascade zone represented by the first conjunction of formulus (1), that is:

$$(x_1 - x_2 \leq -2 \wedge x_2 - x_4 \leq -3 \wedge x_4 - x_1 \leq 5 \wedge x_1 = x_3)$$

Given a zone ζ, we use ζ^S to denote the cascade form of ζ. The procedure to compute ζ^S from ζ can be presented as the following steps.

1 We classify the clocks in a state into equivalence classes, just as what has been done in [14]. Two clocks x, x' are in the same equivalence classes in a zone ζ iff $x - x' = d \in \zeta^C$ for some $d \in N$. ($x - x' = d \in \zeta^C$ is a shorthand for $x - x' \leq d \in \zeta^C \wedge x' - x \leq -d \in \zeta^C$.) For example, in the first conjunction of formulus (1), all four clocks are in the same equivalence class; while in the second conjunction, only clock x_1, x_2, x_4 are in the same equivalence class.

In the following, then we divide the task into two subtasks. The first task is for adding inequalities in ζ^S between those clocks in the same equivalence classes while the second task is for adding inequalities in ζ^S between those clocks in different classes.

2 **Case of clocks in the same equivalence class:** Suppose we are given an equivalence class of clocks in a zone ζ.

(a) We first classify clocks in the given equivalence class into identity classes. Two clocks are in the same identity class if they have the same reading. For example, in the first conjunction of formulus (1), only clocks x_1, x_3 are in the same identity class and all other two clocks are not in the same identity class. But in the second conjunction, no two clocks are in the same identity class.

(b) Then we arrange the identity classes in a linear sequence according to their less-than relation. For example, in the first conjunction of formulus (1), we have the following sequence.

$$\{x_1, x_3\} \xrightarrow{-2} \{x_2\} \xrightarrow{-3} \{x_4\}$$

Here the label of –2 is on the first "$\rightarrow$" because $x_1 - x_2 \leq -2$ and $x_3 - x_2 \leq -2$. But in the second conjunction, the sequence for the equivalence class $\{x_1, x_2, x_4\}$ is

$$\{x_1\} \xrightarrow{-2} \{x_2\} \xrightarrow{-3} \{x_4\}$$

(c) For each two clocks x, x' in the same identity class, let $\zeta^S(x, x') = (\leq, 0)$.

(d) For two clocks x, x' such that the identity class of x just precedes that of of x' in the linear sequence, let $\zeta^S(x, x') = \zeta^C(x, x')$.

(e) For a clock x in the last identity class and a clock x' in the first identity class, let $\zeta^S(x, x') = \zeta^C(x, x')$.

3 **Case of clocks not in the same equivalence class:** Suppose we are given clock x, x' not in the same equivalence class. If x is in the last identity

class of its equivalence class and x' is in the first identity class of its equivalence class, then let $\zeta^S(x, x') = \zeta^C(x, x')$.

4 For any two inequivalent clocks x, x' such that (x, x') is redundant in ζ, we let $\zeta^S(x, x') = (<, \infty)$.

5 For all other clock difference relation not covered in the above-mentioned steps, we let $\zeta^S(x, x') = (<, \infty)$.

In the same style of the manipulation algorithm presented in [17], we can implement the above-mentioned procedure with symbolic CRD+BDD manipulation routines. But due to page-limit, we shall only present the above procedure. After CRD are stored with zones in cascade form (see for example in figure 2(c)), we can then use the reduction operation mentioned at the end of subsection 3.2.3 to efficiently eliminate many contained zones (see for example in figure 2(d)).

One good property of our cascade CRD is that for a zone ζ wihtout non-trivial identity classes (a class is trivial if it has only one element), then the corresponding zone in cascade form have the same number of inequalities as its counterpart in reduced form.

LEMMA 1 : Given a zone ζ such that no two clocks are in the same identity class in ζ, then $\zeta^S$and ζ^R have the same number of inequalities of the form $x - x' \sim d$ with either "$\sim$"$\neq$"$<$" or $d \neq \infty$.

Proof : When there is no nontrivial identity class, we will skip step 2(c) in the our procedure. This skip makes our procedure essentially identical to the one for constructing zones in reduced form presented in [14], except for the following. For an equivalence class with clock x and without clock x', an inequality like $x - x' \sim d$ is called an incoming arc while one like $x' - x \sim d$ is called an outgoing arc. In the zones in reduced form constructed according to [14], only incoming and outgoing arcs to the clocks in the first trivial identity class in the sequence can be kept. But in our cascade zones, outgoing arcs are recorded for the clocks in the first trivial identity class in the sequence while incoming arcs are recorded for the clocks in the last trivial identity class in the sequence. With this way of accounting, it is easy that both cascade form and reduced form of the same zone have the same number of clock inequalities. ||

Lemma 1 shows that our cascade CRD only adds to complexity when it is making effect on its target zones.

5. IMPLEMENTATION AND EXPERIMENTS

We have implemented our CRD-technology in version 3.0 of our tool red which was previously announced in [15, 16] (version 1.0, 2.0) and supports the modelling and safety-analysis of real-time systems with multiprocesses, pointer data-structures, and synchronizations (synchronous send and receive)

from one process to another. The new version, together with benchmarks, is now available at:

`http://www.iis.sinica.edu.tw/~farn/red`

Each process can use global and local variables of type clock, discrete, and pointer. Pointer variables either contain value NULL or the identifiers of processes. Thus in the models input to `red`, we allow complicate dynamic networks to be constructed with pointers.

At this moment, `red` supports backward reachability analysis. We have also implemented a reduction techniques in `red`. That is the reduction by elimination of inactive variables[13, 18] which is always executed. A variable is inactive in a state iff it is not read in any computation from the state before its content is overwritten. Contents of inactive variables can be omitted from state information without any effect on the computations.

We have tested our verifier with four implementations of the web-camera system, each with only one client. We have cooperated with Metamedia, a local company specializing in driver and peripheral software (`http: //www.mmedia.com.tw`), to test our new CRD-technology. One of their current projects is web cameras with browser interface (HTML language). The product supports multi-user connections, with web browsers, to dynamically monitor remote activities through internet. The software operates in application layer with an event queue and software interrupt handling. We want to analyze how long the event queue buffer needs to be.

In the following, we shall first describe how we model the system behavior, and then present our analysis result in section 5. There are three parameters in the system: number of clients (in symbols $\#C$), maximal length of queue (L), and the communication delay between server and clients $[\beta,\alpha]$. We have modeled implementations of the system with various values of $\#C$, L, and $[\beta, \alpha]$. The total number of processes we need is $3 + 2\#C$. The processes are described in the following.

- The event queue which supports operations of dequeue and enqueue; and signalling of new event in the queue to the server. We need variables $e_0, e_1, \ldots, e_{L-1}$ to records the content of the queue; variable l to records the current length of the queue.
- The camera process is also modeled by the single-mode automaton which outputs a video image every 20 time units. The video-ready event will be caught by the queue process throught a synchronizer.
- $\#C$ timer processes for communication respectively with $\#C$ clients. A timer process signals the first timeouts when its corresponding client does not respond in 40 time units. Then the server will send a new frame to the client. If still no response is received in 30 time units, the channel is disconncted.

- *#C* client processes which acknowledges within time interval [β, α] the reception of a frame or dies.
- The server process which processes the events in the queue in the following way.
  - When the event is "VIDEO_READY" by camera, the server sends out a video-frame to each client. The processing time per client per video-frame is 5 time units. After processing the "VIDEO-READY" event, a timer is activated with timeout value 40 time units for each client.
  - When the event is "TIME_OUT_j^1" for client *j* at 40 time units, the server sends out a video-frame to client *j* again (in 5-time-unit processing time) and activates a timer with timeout value 30 time units.
  - When the event is "TIME_OUT_j^2" for client *j* at 30 time units, then the server thinks that the client is dead and disconnect the service to client *i.*
  - When the event is "ACK_j" (acknowledgment) from client *i,* then it disables the timer activated for client *i.*

Two variables, each with possible values, orthogonally generate the four implementations. The first variable is *L,* the event queue maximal length, with possible values of 1 and 2. The second variable is the response time interval from the client [β, α] with possible intervals of [1, 1] and [20,40]. The property to verify is whether the queue will overflow or not.

With reduced CRD, our tool cannot finish the verification task in 6 hours for any of the four implementations. But with cascade CRD, our tool is capable of coming up with the following performance data.

	L = 1	*L = 2*
[β,α] = [1, 1]	no overflow/584s/812k	no overflow/2946s/3257k
[β, α] = [20, 40]	overflow/2320s/l378k	no overflow/3072s/2829k

s: seconds; k: kilobytes of memory in data-structure;

The performance data is collected on a Pentium II 366MHz with 256MB memory running Linux.

6. CONCLUSION

CRD gains its power primarily from intensive data-sharing. But its manipulations are usually burdened with high overhead. At this moment, CRD-technology only shows its edge when number of clocks is large (see the appendix). But we believe that as a new data-structure for the verification of timed automata, there can be much room for improvement on CRD. Especially, we will try out various zone forms and verification techniques appropriate for CRD.

REFERENCES

[1] Asarin, Bozga, Kerbrat, Maler, Pnueli, Rasse. Data-Structures for the Verification of Timed Automata. Proceedings, HART'97, LNCS 1201.

[2] R. Alur, C. Courcoubetis, D.L. Dill. Model Checking for Real-Time Systems, IEEE LICS, 1990.

[3] F. Balarin. Approximate Reachability Analysis of Timed Automata. IEEE RTSS, 1996.

[4] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L.Dill, L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond, IEEE LICS, 1990.

[5] M. Bozga, C. Daws. O. Maler. Kronos: A model-checking tool for real-time systems. 10th CAV, June/July 1998, LNCS 1427, Springer-Verlag.

[6] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, Wang Yi. UPPAAL - a Tool Suite for Automatic Verification of Real-Time Systems. Hybrid Control System Symposium, 1996, LNCS, Springer-Verlag.

[7] G. Behrmann, K.G. Larsen, J. Pearson, C. Weise, Wang Yi. Efficient Timed Reachability Analysis Using Clock Difference Diagrams. CAV'99, July, Trento, Italy, LNCS 1633, Springer-Verlag.

[8] R.E. Bryant. Graph-based Algorithms for Boolean Function Manipulation, IEEE Trans. Comput., C-35(8), 1986.

[9] E. Clarke, O. Grumberg, M. Minea, D. Peled. State-Space Reduction using Partial-Ordering Techniques, STTT 2(3), 1999, pp.279-287.

[10] D.L. Dill. Timing Assumptions and Verification of Finite-state Concurrent Systems. CAV'89, LNCS 407, Springer-Verlag.

[11] C. Daws, A. Olivero, S. Tripakis, S. Yovine. The tool KRONOS. The 3rd Hybrid Systems, 1996, LNCS 1066, Springer-Verlag.

[12] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine. Symbolic Model Checking for Real-Time Systems, IEEE LICS 1992.

[13] P.-A. Hsiung, F. Wang. User-Friendly Verification. Proceedings of 1999 FORTE/PSTV, October, 1999, Beijing. Formal Methods for Protocol Engineering and Distributed Systems, editors: J. Wu, S.T. Chanson, Q. Gao; Kluwer Academic Publishers.

[14] K.G. Larsen, F. Larsson, P. Pettersson, Y. Wang. Efficient Verification of Real-Time Systems: Compact Data-Structure and State-Space Reduction. IEEE RTSS, 1998.

[15] F. Wang. Efficient Data-Structure for Fully Symbolic Verification of Real-Time Software Systems. TACAS'2000, March, Berlin, Germany in LNCS, Springer-Verlag.

[16] F. Wang. Region Encoding Diagram for Fully Symbolic Verification of Real-Time Systems. the 24th COMPSAC, Oct. 2000, Taipei, Taiwan, ROC, IEEE press.

[17] F. Wang. Clock Restriction Diagram: Yet Another Data-Structure for Fully Symbolic Verification of Timed Automata, Technical Report TR-IIS-01-002, IIS, Academia Sinica, Taiwan, ROC.

[18] F. Wang, P.-A. Hsiung. Automatic Verification on the Large. Proceedings of the 3rd IEEE HASE, November 1998.

[19] S. Yovine. Kronos: A Verification Tool for Real-Time Systems. International Journal of Software Tools for Technology Transfer, Vol. 1, Nr. 1/2, October 1997.

Appendix: Performance comparison between our CRD-based tool and Kronos and UPPAAL

The following table, from [17], shows performance data from Kronos, UPPAAL, and **red** w.r.t. implementations of benchmarks with various numbers of concurrent processes. Fischer's mutual exclusion protocol are modified from [3, 13, 15, 18]. CSMA/CD is extracted from [19] while FDDI is from [5, 11]. Information on the three benchmarks can be found in **http://www.iis.sinica.edu.tw/~farn/red.**

benchmarks	concurrency	Kronos	UPPAAL	**red** no CRD	**red** (w. CRD)
Fischer's mutual exclusion	3 processes	0.03s	0.014s	45.42s/1448k	1.12s/48k
	4 processes	0.14s	0.197s	871.2s/12703k	8.09s/162k
	5 processes	0.989s	5.94s	O/M	48.6s/471k
	6 processes	O/M	537.7s	O/M	278s/1271k
	7 processes	O/M	O/M	O/M	1593s/3208k
	8 processes	O/M	O/M	O/M	9352s/7788k
CSMA/CD	3 processes	0.032s	0.0046s	O/M	0.89s/60k
	4 processes	0.071s	0.028s	O/M	3.63s/121k
	5 processes	0.309s	0.216s	O/M	13.8s/278k
	6 processes	1.915s	3.45s	O/M	60.6s/707k
	7 processes	O/M	172s	O/M	227s/1781k
	8 processes	O/M	O/M	O/M	709s/4426k
	9 processes	O/M	O/M	O/M	3580s/10851k
FDDI token-ring passing	11 stations	399s	0.34s	N/A	1.98s/502k
	12 stations	O/M	0.487s	N/A	2.90s/591k
	20 stations	O/M	4.05s	N/A	27.28s/1676k
	30 stations	O/M	25.27s	N/A	151s/4165k
	40 stations	O/M	95.99s	N/A	465s/7939k
	50 stations	O/M	O/M	N/A	1107s/13513k
	60 stations	O/M	O/M	N/A	1828s/21014k

data collected on a Pentium III 800MHz with 256MB memory running LINUX;
s: seconds; k: kilobytes of memory in data-structure; O/M: Out of memory; N/A: not available;

For the Fischer's and CSMA/CD benchmarks, UPPAAL is invoked with options "`-aSWD`." The performance data shows that CRD-technology is still more space-efficient and scales better w.r.t. number of clocks. For the FDDI benchmark, UPPAAL is invoked with options "`-STDda`" where "`d`" is for depth-first-search and suits very well for highly synchronous algorithms like FDDI.

We remind the readers that compared to those well-established tools like Kronos and UPPAAL. For example, UPPAAL can pass most of the benchmarks in very short time if the option of "convex-hull over-approximation" is switched on. But since our experiment was focused to argue that "with similar and equivalent verification techniques, BDD-like data-structures can outperform DBM," we think it is better to just compare with Kronos and UPPAAL in the context of precise verification. Definitely, the success experience of Kronos and UPPAAL will be the guide for future enhancement of **red.** In the future, when such techniques are incorporated with CRD-based technology, we believe there will be much room for performance enhancement.