

Adding real time into state machine analysis of digital evidence

Pavel Gladyshev
School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland

Technical Report UCD-CSI-2006-3

Abstract

This report describes an extension of the finite state machine theory of digital event reconstruction. The proposed extension adds the known times of witness observations as formal objects in the theory and uses them to compute temporal bounds of events, whose time is not known.

1 Introduction

State machine analysis of digital evidence is a rigorous method for determining possible sequences of events that could have happened in a digital system during an incident. The method is based on modelling the system under investigation as a state machine and exploring its possible computation space as described in [2,3,4]. The main idea of the method can be summarised as follows: the system under investigation is modelled as a finite state machine; suppose that after the incident it has been found in some state x . Possible sequences of events that could have happened in the system during the incident can be determined by

1. backtracing transitions leading to the final state x ;
2. discarding sequences of transitions that disagree with the available evidence.

A formalisation of this idea along with the corresponding event reconstruction algorithm was given in [2,3]. The key points of this formalisation are summarised in Section 2.

1.1 Contribution of this paper

The formalisation of event reconstruction given in [2,3] does not provide a way to reason about real world times of events that happened during the incident. This is unfortunate, because the knowledge of time of events can be vital to the success or failure of investigation.

One of the ways to address this limitation is to extend the model of [2,3] with “event time bounding” reasoning, which refers to the following argument: Suppose that there are three events A, B, and C. If event A caused event B and event B

caused event C, then the time of B must be between the times of A and C. A general algorithm for this kind of reasoning has been published in [5]. The reader is referred to that publication for more information and examples.

This report presents formal developments required for incorporation of event time bounding reasoning into the state machine model of digital event reconstruction.

2 State machine formalisation of event reconstruction

This section gives formal definition of event reconstruction problem defined in [2]. This definition is repeated here because it serves as the basis for formal developments given in Sections 3 and 4.

In summary, the system under investigation is modelled as a *finite state machine*. The available evidence is modelled as the *evidential statement*, which expresses the evidence as a collection of witness observation about the state and change of observable system properties during the incident. The *event reconstruction problem* is then defined as finding all possible explanations for the given evidential statement with respect to the given finite state machine.

2.1 Mathematical Notation

Sets are denoted by capital Roman letters A, B, C, \dots . The set of integers is denoted \mathbb{Z} , and the empty set is denoted \emptyset .

Sets are defined either by explicit enumeration $A = \{1, 2\}$ or by a set former $A = \{a \mid p(a)\}$.

Set product of A and B is denoted $A \times B$. The n^{th} power of set A is denoted A^n . The powerset (the set of all subsets) of set A is denoted 2^A .

Symbols $\subset, \subseteq, \supset, \supseteq, \cup, \cap, \in$, and \notin are used in their usual mathematical meaning.

Sequences are denoted by lower case letters a, b, c, \dots

Sequences are defined by listing their elements $s = (a, b, c)$. Empty sequence is denoted ε .

The length of sequence s is denoted $|s|$.

Elements of sequence s are denoted s_i , where i is an integer $0 \leq i < |s|$.

Head of sequence s is its first element s_0 . Tail of sequence s is the sequence $(s_1, \dots, s_{|s|-1})$.

Concatenation of sequences s and q is denoted $s \cdot q$.

Function δ that maps set A into set B is denoted $\delta: A \rightarrow B$.

Subscripts and superscripts are also used for specifying relationship between two objects. For example, C_T denotes the set of all finite computations of the finite state machine T is denoted C_T . Such use will be obvious from the context.

2.2 Finite state machine

For the purposes of this report, *finite state machine* is defined to be a tuple of three elements $T = (Q, I, \phi)$, where

- I is a finite set of all possible events,
- Q is a finite set of all possible states,
- $\phi: I \times Q \rightarrow Q$ is a transition function that determines the next state for every possible combination of event and state.

Transition is defined to be the process of state change. Transitions are instantaneous.

A (finite) *computation* is defined to be a non-empty, finite sequence of steps, where each step is a pair $c_j = (c^{t_j}, c^{q_j})$, where

$c^{t_j} \in I$ is event, $c^{q_j} \in Q$ is a state, and any two steps c_k and c_{k-1} are related via transition function

$$\text{for all } 1 \leq k < |c|, c^{q_k} = \phi(c^{t_{k-1}}, c^{q_{k-1}})$$

The *set of all finite computations* of the finite state machine T is denoted C_T .

2.3 Run

The concept of run is introduced for formal convenience. It can be seen as an "extension" of the concept of computation.

Like computation, the run describes the evolution of system state over time. However, in addition to the current state and event, each element of the run contains all "future" states and events that will happen in the computation after the current state.

This unusual view of computation allows formalisation of witness observations that relate several consecutive system states, and it also gives convenient basis for formalisation of transition back-tracing.

Formally, a *run* is a (possibly empty) sequence of finite computations, in which the next computation is obtained from the previous computation by discarding its first state-event pair:

$$\text{for all } 1 \leq i < |r|, r_i = \phi(r_{i-1})$$

the purpose of function ϕ here is to discards the first element of the given computation. Formally function ϕ is defined as follows: For two computations $x \in C_T$ and $y \in C_T$, $y = \phi(x)$ if and only if $x = (x_0) \cdot y$.

The *set of all runs* of the finite state machine T is denoted R_T .

The *run of computation* c is defined to be a run whose first computation is c .

2.3.1 Infinitum

The *infinitum* is another object introduced for formal convenience. It is defined as an integer constant that is greater than the length of any computation that may have happened during the incident.

Intuitively, the *infinitum* stands for an arbitrary amount of time. It is used in formal evidential statements to specify that the duration of the witness observation is unknown. At the same time, *infinitum* has been defined as a (very large but finite) constant to ensure the termination of the event reconstruction algorithm defined in [2,3].

2.4 Partitioned run

To associate witness observations with segments of computation, the concept of partitioned run is defined.

Partitioned run is defined to be a finite sequence of runs $pr \in (R_T)^{|pr|}$, such that concatenation of its elements in the order of listing is also a run:

$$(pr_0 \cdot pr_1 \cdot \dots \cdot pr_{|pr|-1}) \in R_T$$

The *set of all partitioned runs* of the finite state machine T is denoted PR_T .

A *partitioning* of run r is defined to be a partitioned run, denoted pr_r , concatenation whose elements produces r :

$$pr_{r_0} \cdot pr_{r_1} \cdot \dots \cdot pr_{r(|pr_r|-1)} = r$$

2.5 Formalisation of evidence

Informally speaking, the evidence is formalised as a set of witness “stories” about the state and change of observable system properties during the incident. The event reconstruction is seen as the job of finding all runs of the state machine that do not contradict any of the witness stories.

2.5.1 Observation

Observation is a formal object that represents a witness statement that system behaviour exhibited some observable (to the witness) property *continuously for some time*.

Syntactically, observation is defined to be a triple

$$o = (P, \min, \text{opt}),$$

where P is the set of all computations of T that possess the property observed by the witness, and \min and opt are non-negative integers that restrict duration of observation.

An *explanation* of the observation o is defined to be a run r such that

- $r_i \in P$ for all $0 \leq i < |r|$, and
- $\min \leq |r| \leq \min + \text{opt}$

The *meaning* of observation o is defined to be the set R_o of all runs that explain observation o .

2.5.2 Observation sequence

To make sense of individual observations, these observations need to be arranged in sequence to make a story.

An *observation sequence* is defined to be a non-empty sequence of observations listed in chronological order:

$$os = (o_A, o_B, o_C, \dots)$$

An *explanation of observation sequence* os is defined to be a partitioned run pr such that $|pr| = |os|$ and each element of pr is an explanation of the corresponding observation in os .

The *meaning of observation sequence* os is defined to be the set $PR_{os} \subseteq PR_T$ of all partitioned runs that explain os .

A run r is said to *satisfy* an observation sequence os if and only if there exists a partitioning of r that explains os .

A computation c is said to *satisfy* an observation sequence os if and only if the run of c satisfies os .

2.5.3 Evidential statement

If there was more than one witness of the incident, stories of all the witness must be considered when assessing possibility of a particular incident scenario. The concept of evidential statement is defined to group witness stories (observation sequences) together.

An *evidential statement* is defined to be a non-empty sequence of observation sequences

$$es = (os_X, os_Y, os_Z, \dots)$$

Evidential statement combines restrictions imposed by all of its observation sequences.

An *explanation* of evidential statement es is defined to be a sequence of partitioned runs spr , such that

- all elements of spr are partitionings of the same run r ,
- each element of spr is an explanation of the corresponding observation sequence of es , and
- The number of elements in spr is the same as the number of elements (observation sequences) in es :
 $|spr| = |es|$.

The *meaning of evidential statement* es is the set of all sequences of partitioned runs SPR_{es} that explain es .

Evidential statement is said to be *inconsistent* if it has no explanations.

2.6 Event reconstruction problem

Event reconstruction problem is defined as the task of finding the set of all possible explanations¹ SPR_{es} of the given evidential statement es with respect to the given finite state machine T .

3 Assigning real-world times to transitions

The state machine model defined above represents the time indirectly, as the sequence of state transitions. To allow reasoning about real times of events, the transitions must be linked to their real times during the incident.

As shown in Figure 1, any run r of the state machine T can be associated with $|r| + 1$ state

¹ It can be argued that only a subset of SPR_{es} is useful, whose explanations partition runs not longer than *infinitum*

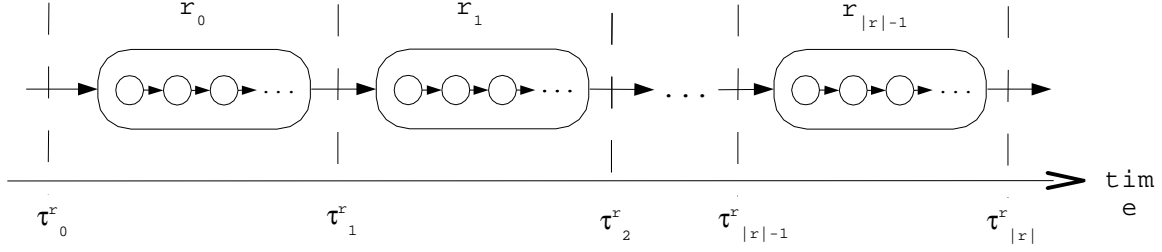


Figure 1. Times of transitions

transitions. There are $|r|$ transitions *into* each element of r and one transition *out* of the last element of r .

3.1 Transition times of a run

Every transition can be associated with a moment in real time at which it happened during the incident.

The *sequence of transition times* of a run r is denoted τ^r . It consists of $|r|+1$ real valued numbers that represent times of every transition in the run. The relationship between elements of τ^r and elements of r is shown in Figure 1.

The first element of τ (τ_0^r) represents the beginning time of the run r . It is the time of the transition *into* the first element of r (r_0).

The last element of τ ($\tau_{|r|}^r$) represents the ending time of the run r . It is the time of transition *out* of the last element of r ($r_{|r|-1}$).

An intermediate i^{th} element of τ (τ_i^r) represents the time of transition from r_{i-1} to r_i .

Elements of τ^r are ordered in time:

$$\tau_i^r < \tau_{i+1}^r, \text{ for all } 0 \leq i < |r| \quad (1)$$

3.2 Temporal precedence of runs

A run ra precedes run rb in time, if the ending time of ra is less than or equal to the beginning time of rb :

$$\tau_{|ra|}^{ra} \leq \tau_0^{rb}$$

If ra and rb are sub-runs of some run $\setminus(rc)$, then positions of ra and rb in rc can be used to determine temporal precedence between ra and rb .

Let i be the index of the first element of ra in rc , and let j be the index of the first element of rb in rc , then

$$\tau_{|ra|}^{ra} = \tau_{i+|ra|}^{rc} \quad \text{and} \quad \tau_0^{rb} = \tau_j^{rc}$$

Run ra precedes run rb if

$$\tau_{|ra|}^{ra} \leq \tau_0^{rb}$$

or, equally

$$\tau_{i+|ra|}^{rc} \leq \tau_j^{rc} \quad (2)$$

which is true if and only if

$$i + |ra| \leq j \quad (3)$$

3.3 Times of observations

Witness observations regarding time of events are formalised as *known times of observations*.

For the purpose of defining known times of observation, the notion of observation identifier is introduced. An *observation identifier* is a pair $id = (i, j)$. It denotes observation $es_{i,j}$ at the j^{th} position of the i^{th} observation sequence of evidential statement es .

A *known time of observation* is defined to a pair $t = (id, tim)$, where $id = (i, j)$ is an observation identifier and tim is a real valued number that represents time. The meaning of t is an assertion that for any run r explaining observation $es_{i,j}$, the following inequality holds

$$\tau_0^r \leq tim \leq \tau_{|r|}^r \quad (4)$$

A known time of observation corresponds to a witness statement that moment tim happened during the witness's observation $es_{i,j}$. Such a statement may result from human looking at a clock during observation, or from an operating system appending clock reading to a log file entry.

4 Determining times of observations by time bounding

To illustrate the use of the formal machinery defined above; this section uses it to automate a well known analysis technique, which is sometimes called “event time bounding”. It refers to the following argument. Suppose that event A caused event B and event B caused event C. Then the time of event B has to be in between times of events A and C. This type of reasoning is useful for disproving suspect alibis and for sanity-checking witness statements.

In this section, the idea of event time bounding is extended to observations. It is shown how to determine time boundaries for a given observation $es_{i,j}$ using known times of other observations.

4.1 Happened before relation between observations

In the state machine model of incident, the causal connections between events are represented indirectly. They are embodied into the transition function. The direct analysis of transition function may be inconvenient. Fortunately, the same information can be found in the set SPR_{es} of all explanations of the given evidential statement es .

In particular, it is possible to define “happened-before” relation between observations, as follows:

An observation $es_{i,j}$ happened before observation $es_{k,l}$ if and only if in every sequence of partitioned runs explaining es the run explaining observation $es_{i,j}$ precedes the run explaining $es_{k,l}$.

The precedence of runs is formally defined by inequalities (2,3).

4.2 Observation time bounding

With the above definition of “happened-before” relation between observations, the event-time bounding of observations can also be defined after [5] as follows:

Consider three observations A, B, and C, possibly in different observation sequences of the same evidential statement. Suppose that A happened before B and B happened before C. Then observation B is bounded between the latest observation time of A and the earliest observation time of C.

The algorithm for computing time bounds of a given observation $es_{i,j}$ is given below.

1. O_{all} = set of all observation identifiers in es
2. $O_{bef} \leftarrow O_{all}$
3. $O_{aft} \leftarrow O_{all}$
4. for every sequence of partitioned runs spr in SPR_{es}
 - 4.1 Compute the set O_{bef}^{spr} of identifiers of all observations whose explaining runs precede the run explaining observation $e_{i,j}$ in spr .
 - 4.2 Compute the set O_{aft}^{spr} of identifiers of all observations whose explaining runs follow² the run explaining observation $e_{i,j}$ in spr .
 - 4.3 $O_{bef} \leftarrow O_{bef} \cap O_{bef}^{spr}$
 - 4.4 $O_{aft} \leftarrow O_{aft} \cap O_{aft}^{spr}$
5. For the earliest possible time bound of observation $e_{i,j}$ choose the latest known observation time from observations in O_{bef}
6. For the latest possible time bound of observation $e_{i,j}$ choose the earliest known observation time from observations in O_{aft}

Algorithm 1. Observation time bounding.

Steps 1-4 of the algorithm identify all observations whose explanations either precede (O_{bef}) or follow (O_{aft}) the run that explains observation $es_{i,j}$ in all possible explanations of the evidential statement.

Steps 5 and 6 then choose the time bounds of $es_{i,j}$ as the latest known observation time preceding $es_{i,j}$ and the earliest known observation time following $es_{i,j}$.

This algorithm has been implemented as part of EARL package [6].

5 Concluding remarks

This report described formal machinery for associating times with observations in the evidential statement and an algorithm for computing temporal

² i.e. the run explaining $e_{i,j}$ precedes these runs.

bounds of observations on the basis of known times of other observations. However, a number of subtle points remain to be addressed. These are dealt by the subsections below

5.1 A note on reliability of known times of observations

The time bounding algorithm presented in this report assumes that the known times of observations are absolutely reliable. This assumption simplifies reasoning by avoiding reasoning with uncertainty. This assumption is acceptable, because known times can be introduced into analysis gradually. First, time bounding can be performed with only the most reliable known times. If the results of time bounding are unsatisfactory, it can be repeated with less reliable known times included.

Reliability of time bounding results can be improved by checking consistency of known times. All known times of observations must respect happened-before ordering imposed by the evidential statement. This can be checked by (1) calculating time bounds for every observation in the evidential statement and (2) verifying that every known time of observation falls in between the calculated time bounds for its corresponding observation.

5.2 Related work

As discussed in [5], event time bounding is not a universal method for temporal reasoning. A number of other methods have been described in literature.

The ability of web servers to insert timestamps into web pages was utilised in [8]. It was shown that a web page stored in a web browser's disk cache has two timestamps. One timestamp is the creation time of the file, which contains the web page. The second timestamp is the timestamp added by the web server. The difference between the two timestamps reflects the skew of the local clock.

A general model for relating clock readings on different computers through a hierarchical system of clock offsets has been described in [7].

The apparatus of computer history models and temporal reasoning based on them has been proposed in [1]. While computer history models also employ state machine models and assign times to model transitions, the event time bounding is not directly addressed in [1].

Most recently, an experimental study of clock drift has been reported in [9]. It demonstrates that timestamps produced by the computer maybe unreliable during periods of clock synchronization.

6 References

1. Carrier B. *Hypothesis Based Approach to Digital Forensic Investigation*. PhD Dissertation; CERIAS; Purdue University; 2006.
2. Gladyshev P. *Formalising event reconstruction in digital investigations*. PhD Dissertation; University College Dublin; 2004
3. Gladyshev P., Patel A. Finite state machine approach to digital event reconstruction. *Digital Investigation Journal* 2004; 1(2)
4. Gladyshev P. Finite state machine analysis of a blackmail investigation. *International Journal of Digital Evidence* 2005; 4(1)
5. Gladyshev P., Patel A. Formalising Event Time Bounding in Digital Investigations. *International Journal of Digital Evidence* 2005; 4(2)
6. Event Analysis and Reconstruction in Lisp (EARL) software package. 2005
<http://www.glayshev.info/smforensics/earl>
7. Stevens M. Unification of relative time frames for digital forensics. *Digital Investigation journal, vol. 1, issue 3, 2004, pp. 255-239*
8. Weil M. Dynamic Time & Date Stamp Analysis. *International Journal of Digital Evidence*; 1(2); 2002.
9. Schatz B., Mohay G., and Clark A. A correlation method for establishing provenance of timestamps in digital evidence. *In Proceedings of Digital Forensics Research Workshop (DFRWS 2006), Lafayette, Indiana; 2006*

