

# Using Simplified Event Calculus in Digital Investigation

Svein Yngvar Willassen  
Department of Telematics  
Norwegian University of Science and Technology  
O.S. Bragstads plass 2  
7491 Trondheim, Norway  
+47 92449678  
svein@willassen.no

## ABSTRACT

In a hypothesis-based approach to digital investigation, the investigator formulates his hypothesis about which events took place, and tests them using the evidence available. A formalism for the description of the investigated system is useful in the hypothesis formulation and testing. Simplified Event Calculus, a form of propositional logic, can be used to define and test hypotheses in a digital investigation. When a system is modelled in this logic, observed states can be used to find action hypotheses and test them in the model. This can assist investigators and fact-finders in reconstruction of events from digital evidence. The logic can also be used to derive invariants for a system that can be utilized in tools checking evidence from these systems for consistency.

## Categories and Subject Descriptors

F.4.1 [Mathematical Logic and formal languages]:  
Mathematical Logic – *model theory, temporal logic.*

## General Terms

Theory, Legal Aspects, Verification.

## Keywords

event calculus, digital investigation, propositional logic

## 1. INTRODUCTION

Investigations are inquiries into past events. The purpose of an investigation is to find evidence that can establish an understanding of events previously taken place. Investigation of digital media with the purpose of finding evidence is commonly referred to as *digital investigation*. The purpose of digital investigation is to find evidence related to the events under investigation. In recent works, most notably by Carrier, efforts have been made to make the digital investigation process based on scientific principles, by using a hypothesis-based approach. [1] In this approach, the investigator formulates his hypothesis about

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'08, March 16-20, 2008, Fortaleza, Cear, Brazil.  
Copyright 2008 ACM 978-1-59593-753-7/08/0003...\$5.00.

which events took place, and tests them using the available evidence.

In a hypothesis-based approach, it is useful to use a formalism to describe the system under investigation, and the events that have taken place. Such a formalism needs to be able to describe events occurring on a system and their effect on the state, in such a way that the investigator's hypothesis can be tested for consistency with the evidence on the examined system. Previous works have used variants of Finite State Machines to represent the system under investigation and parts thereof. [1, 2] In this work, a variant of propositional logic will be used for the same purpose. This paper examines if and how Shanahan's Simplified Event Calculus [3] can be used in a hypothesis based approach to digital investigation.

## 2. SIMPLIFIED EVENT CALCULUS

In Simplified Event Calculus, the world is modelled with *fluents* and *actions*. Fluents are states that can hold for a specified or unspecified period of time. Actions are occurrences that initiate or terminate a fluent. Occurrences of actions and fluents are defined with the *HoldsAt* and *Happens* predicates, and the affection of actions on fluents is defined by the predicates *Initiates* and *Terminates*. In addition, there is an *Initially* predicate, for initiating fluents from the start. The effect axioms of the Simplified Event Calculus are:

$$\text{HoldsAt}(f,t2) \leftarrow \text{Happens}(a,t1) \quad (1)$$
$$\wedge \text{Initiates}(a,f,t1) \wedge t1 < t2 \wedge \text{not Clipped}(t1,f,t2)$$

$$\text{Clipped}(t1,f,t2) \leftarrow \quad (2)$$
$$\text{Happens}(a,t) \wedge \text{Terminates}(a,f,t) \wedge t1 < t < t2$$

$$\text{HoldsAt}(f,t) \leftarrow \text{Initially}(f) \wedge \text{not Clipped}(0,f,t) \quad (3)$$

**Definition 1.** An *event calculus program* is the conjunction of,

- A finite set S of Initially clauses of the form,  
Initially(f)
- A finite set A of Happens clauses of the form,  
Happens(a, t)
- A finite set E of Initiates clauses and a finite set of Terminates clauses of the form,  
Initiates(a, f1, t)  $\leftarrow$   $\Pi$   
Terminates(a, f1, t)  $\leftarrow$   $\Pi$

where  $\Pi$  does not mention the predicates Initially, Happens, Initiates or Terminates and every occurrence of the HoldsAt predicate is of the form

$$\text{HoldsAt}(f_2, t)$$

- The effect axioms of simplified event calculus
- A finite set of general clauses not mentioning the predicates Initially, Happens, Initiates, Terminates or  $<$ .

For further description of the Simplified Event Calculus, the reader is referred to Shanahan's work on the subject. [3]

### 3. A SIMPLE FILE SYSTEM MODEL

For the purpose of this paper, we define a simple file system: This file system contains files, and each file has an Accessed timestamp and a Modified timestamp. Files can be Read or Written. Reading a file causes the Accessed timestamp to be updated. Writing a file causes both the Accessed timestamp and the Modified timestamp to be updated. There is only one timestamp of each type for each file, so whenever a timestamp is changed, the previous value is lost.

Changes in the simple file system can now be represented as an event calculus program, where the fluents are file timestamps with an associated clock value, and actions are the operations that might change the timestamps. For the simple file system, the fluents can be called Accessed(file,  $\tau_a$ ) and Modified(file,  $\tau_m$ ). Further, actions can be represented with Read(file) and Write(file). The set E of Initiates and Terminates clauses will then contain clauses that Initiates timestamps with the value from current system clock  $c(t)$ , and Terminates them:

$$\text{Initiates}(\text{Read}(\text{file}), \text{Accessed}(\text{file}, c(t)), t) \quad (4)$$

$$\text{Initiates}(\text{Write}(\text{file}), \text{Accessed}(\text{file}, c(t)), t) \quad (5)$$

$$\text{Initiates}(\text{Write}(\text{file}), \text{Modified}(\text{file}, c(t)), t) \quad (6)$$

$$\text{Terminates}(\text{Read}(\text{file}), \text{Accessed}(\text{file}, c(t_1)), t) \leftarrow t_1 < t \quad (7)$$

$$\text{Terminates}(\text{Write}(\text{file}), \text{Accessed}(\text{file}, c(t_1)), t) \leftarrow t_1 < t \quad (8)$$

$$\text{Terminates}(\text{Write}(\text{file}), \text{Modified}(\text{file}, c(t_1)), t) \leftarrow t_1 < t \quad (9)$$

In most real file systems there is always a value assigned to the time stamps of a file. It therefore makes sense to define Initially clauses that initiates fluents for the timestamps, so they will hold from the start:

$$\text{Initially}(\text{Accessed}(\text{file}, \tau_0)) \quad (10)$$

$$\text{Initially}(\text{Modified}(\text{file}, \tau_0)) \quad (11)$$

In the simple file system model, S is the conjunction of formulae (10) - (11) and E is the conjunction of formulae (4) - (9). With a definition of a set A of Happens clauses, an event calculus program for this simple file system has been completed. Then, SLDNF resolutions can be utilized to search the space of possible event histories and test propositions about fluents at particular moments in time.

**Example 1.** Let a file be Read at  $t = t_R$  and subsequently written at  $t = t_W$ , so that  $t_R < t_W$ . Let  $c(t)$  be an integer, so that  $\tau_0 = 0$ ,  $c(t_R) = 5$  and  $c(t_W) = 10$ . Let an event program be defined by (4) - (11) and the following clauses in A:

$$\text{Happens}(\text{Read}(\text{file}), t_R)$$

$$\text{Happens}(\text{Write}(\text{file}), t_W)$$

The timestamp fluents at certain moments in time can now be

examined by means of SLDNF resolutions. For example, let us determine if the accessed time stamp at time  $t = t_{\text{Obs}}$ ,  $t_R < t_W < t_{\text{Obs}}$  has value 10.

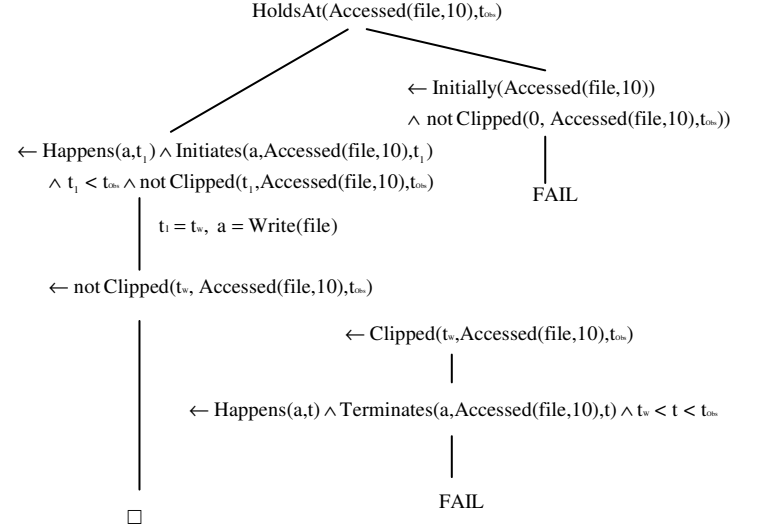


Figure 1. Resolution of HoldsAt(Accessed(file, 10),  $t_{\text{Obs}}$ )

Figure 1 shows a resolution for the observation of the Accessed time stamp given a specific observation time. The right hand branch of the resolution, representing the case that the time stamp was initially set to the observed value fails due to the fact that there is no Initially clause setting the Accessed time stamp to 10. The left hand branch of the resolution assumes that an action happened at time  $t_1$  initiating the fluent Accessed(file, 10) at time  $t_1$ . The only Happens clause that can satisfy this is Happens(Write(file),  $t_W$ ). Since the evaluation of the clause Clipped( $t_W$ , Accessed(file, 10),  $t_{\text{Obs}}$ ) fails, the left branch succeeds and we can conclude that HoldsAt(Accessed(file, 10),  $t_{\text{Obs}}$ ) holds.

### 4. OBSERVATION SETS

The example in the previous section showed how Simplified Event Calculus can be utilized to determine if specific timestamp values holds at a specific moment in time, given known occurrence of actions. This can be extended into taking the final state of the system into account.

**Definition 2.** Formulated in the Simplified Event Calculus, an *Observation Set O* is a finite set of HoldsAt clauses on the form HoldsAt( $f, t_{\text{Obs}}$ )

The *observation proposition* is the conjunction of the HoldsAt clauses in the observation set. The observation proposition has the form

$$p = \varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n$$

Where each  $\varphi$  is a HoldsAt clause contained in O, and n is the number of elements in O.

With the definition of an observation set, the relationship between an observation set and the sets S, E and A defining an event calculus program can be investigated. An event calculus program defines the behaviours occurring in a system in terms of the initial state (S), the effect any actions would have on the states (E) and

the actions that actually occurred (A). With known S, E, and A, possible states at a specific moment in time can be tested for consistency with the event calculus program. When S, E and A are known, SLDNF resolutions can be used to test observation propositions and therefore confirm or refute possible observation sets O. The observation set

$$O = \{\text{HoldsAt}(\text{Accessed}(\text{file},10),t_{\text{obs}})\}$$

was in Example 1 determined to be a possible observation set for S, E and A. This shows how a possible observation set can be tested for consistency with an event calculus program.

Things are however different in an investigation situation. In an investigation, the state at the time of the investigation is observable, whereas information about occurred events is unknown. Under the assumption that the investigator has all information about the initial state S, and also thorough knowledge about the workings of the system, E, the investigator can use the knowledge about the observed state O to derive information about occurred events. In this case A is unknown, whereas S, E and O are known. The investigator can now infer knowledge about A from the observation set O and the detailed knowledge about how the system works, S and E.

Returning to Example 1, if the observed set is

$$O = \{\text{HoldsAt}(\text{Accessed}(\text{file},10),t_{\text{obs}})\}$$

and A is unknown, the investigator can now reason that since (from O) the fluent `Accessed(file, 10)` holds at the time of the observation and since (from S) initially `Accessed(file, 0)`, some action must have occurred that terminated `Accessed(file, 0)` and initiated `Accessed(file, 10)`. From E, the investigator knows that this must have been an action occurring at  $t = t_a$ , where  $c(t_a) = 10$ . The investigator also knows that the action must have been either a Read or a Write action, since (again, from E) these are the only actions that can affect the `Accessed` fluent. The investigator can therefore formulate two hypotheses about occurred actions,  $H_1$  and  $H_2$  where  $c(t_a) = 10$ .

$$H_1 = \{\text{Happens}(\text{Read}(\text{file}),t_a)\}$$

$$H_2 = \{\text{Happens}(\text{Write}(\text{file}),t_a)\}$$

These hypotheses can be tested by SLDNF resolution of the observation proposition for both  $H_1$  and  $H_2$ , and both hypotheses will be accepted.  $H_1$  and  $H_2$  are hypotheses about actions that actually took place. If hypotheses about occurred actions are accepted by an event program resolution, it means that they are possible explanations for the observed set O. The hypotheses do however, even if they are accepted, not imply full knowledge of the set of actions A. Even if only one hypothesis is accepted, it is still in the unknown if there were any actions in A for which there exist no evidence anymore. In Example 1, it could for example be the case that the file was Read at some moment prior to  $t_a$ . The timestamp fluent resulting from this Read would be Terminated by the Read occurring at  $t_a$ , and therefore not be observable at  $t_{\text{obs}}$ .

## 5. ACTION HYPOTHESES

**Definition 3.** An action hypothesis H is a finite set of Happens clauses on the form `Happens(a, t)` derived from an observation set O, given finite sets S and E in an event calculus program.

The acceptance of an action hypothesis means that it is a possible set of actions that can explain the observation set O. In order to be able to deduct possible courses of events from an observation set, we would like to find all possible hypotheses H, given an observation set O and knowledge about the system, represented by S and E.

From Definition 2 the elements of an observation set O are HoldsAt clauses representing the fluents that holds at the time of the observation. The observation proposition to be tested in the event calculus program is the conjunction of these HoldsAt clauses and takes the form

$$p = \varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n \quad (12)$$

where each  $\varphi$  is a HoldsAt clause.

From the effect axioms of the Simplified Event Calculus, these HoldsAt clauses may exist either because they held initially (formula (3)) or because an action occurred that initiated them (formula (1)). There is no other way a HoldsAt clause can come to existence than through formulae (1) or (3). It is therefore possible to find all possible action hypotheses by reasoning on the observation proposition, the Initiates clauses in E and the Initially clauses in S. This reasoning does not have to consider termination of fluents as per the Terminates clause in E, since this will be done by means of SLDNF resolution when each hypothesis is tested for acceptance. The proposition that all fluents in an observation set has been initiated is the conjunction of the initiation of each fluent and takes the form:

$$q = \kappa_1 \wedge \kappa_2 \wedge \dots \wedge \kappa_n \quad (13)$$

where each  $\kappa$  is the initiation of the corresponding  $\varphi$  in the observation proposition p. In the following, this proposition will be called the *initiation proposition*.

A fluent may exist because it held initially or because it was Initiated by a clause in E. There may be more than one Initiates clause in E initiating one particular fluent, and these must all be considered. Written in propositional logic, the initiation of a HoldsAt clause takes the form of a disjunction:

$$\kappa_i = \alpha_{i1} \vee \alpha_{i2} \vee \dots \vee \alpha_{im} \vee \eta_i \quad (14)$$

Where  $\kappa_i$  is the i-th `HoldsAt(f,t2)` clause in q,  $\eta_i$  is an Initially(f) clause, m is the number of `Initiates(a,f,t1)` clauses affecting that fluent and each  $\alpha_i$  is a clause on the form `Happens(a,t1)` where there exists a clause `Initiates(a,f,t1)` in E.

The initiation of the fluents in the observation proposition can now be found by inserting (14) into (13), yielding

$$\begin{aligned} q = & (\alpha_{11} \vee \alpha_{12} \vee \dots \vee \alpha_{1m} \vee \eta_1) \\ & \wedge (\alpha_{21} \vee \alpha_{22} \vee \dots \vee \alpha_{2m} \vee \eta_2) \\ & \wedge \dots \\ & \wedge (\alpha_{n1} \vee \alpha_{n2} \vee \dots \vee \alpha_{nm} \vee \eta_n) \end{aligned}$$

q is a conjunction of disjunctive clauses. By reordering it into a disjunction of conjunctive clauses, a set of action hypotheses will be found, where each of the conjunctive clauses in the disjunction is an action hypothesis H.

Consider an event calculus program with S and E as previously defined and O as defined by the following observation proposition, where  $c(t_m) \neq \tau_0$  and  $c(t_a) \neq \tau_0$ :

$$\begin{aligned}
p &= \text{HoldsAt}(\text{Modified}(\text{file}, c(t_m)), t_{\text{Obs}}) \\
&\wedge \text{HoldsAt}(\text{Accessed}(\text{file}, c(t_a)), t_{\text{Obs}})
\end{aligned} \tag{15}$$

The initiation of these fluents can then be expressed as a conjunction of disjunctive clauses as follows:

$$\begin{aligned}
q &= (\text{Happens}(\text{Write}(\text{file}), t_m) \\
&\vee \text{Initially}(\text{Modified}(\text{file}, c(t_m)))) \\
&\wedge (\text{Happens}(\text{Read}(\text{file}), t_a) \\
&\vee \text{Happens}(\text{Write}(\text{file}), t_a) \\
&\vee \text{Initially}(\text{Accessed}(\text{file}, c(t_a))))
\end{aligned}$$

Since there is no  $\text{Initially}(\text{Modified}(\text{file}, c(t_m)))$  or  $\text{Initially}(\text{Accessed}(\text{file}, c(t_a)))$  in  $S$ , we know that these clauses are false.  $q$  then becomes:

$$\begin{aligned}
q &= \text{Happens}(\text{Write}(\text{file}), t_m) \\
&\wedge (\text{Happens}(\text{Read}(\text{file}), t_a) \\
&\vee \text{Happens}(\text{Write}(\text{file}), t_a))
\end{aligned}$$

Rewritten as a disjunction of conjunctive clauses:

$$\begin{aligned}
q &= \text{Happens}(\text{Write}(\text{file}), t_m) \wedge \text{Happens}(\text{Read}(\text{file}), t_a) \\
&\vee \text{Happens}(\text{Write}(\text{file}), t_m) \wedge \text{Happens}(\text{Write}(\text{file}), t_a)
\end{aligned}$$

So here we obtain two different hypotheses from the fluent initiation:

$$\begin{aligned}
H_1 &= \{ \text{Happens}(\text{Write}(\text{file}), t_m), \text{Happens}(\text{Read}(\text{file}), t_a) \} \\
H_2 &= \{ \text{Happens}(\text{Write}(\text{file}), t_m), \text{Happens}(\text{Write}(\text{file}), t_a) \}
\end{aligned}$$

## 6. DERIVING INVARIANTS

The described methods can be used to test the observation proposition in the general case, and thereby determine properties of the simple file system defined in formulae (4) - (11). The general observation proposition for a file in the simple file system was expressed in (15). Now, if  $c(t_m) \neq \tau_0$  and  $c(t_a) \neq \tau_0$ , there must have occurred actions initiating these fluents. As previously determined, these actions must have been

$$\begin{aligned}
H_1 &= \{ \text{Happens}(\text{Write}(\text{file}), t_m), \text{Happens}(\text{Read}(\text{file}), t_a) \} \\
H_2 &= \{ \text{Happens}(\text{Write}(\text{file}), t_m), \text{Happens}(\text{Write}(\text{file}), t_a) \}
\end{aligned} \tag{16}$$

Now, by investigating the three different cases;  $t_m < t_a$ ,  $t_m = t_a$  and  $t_m > t_a$ , properties of this system can be found.

In the case of  $t_m = t_a$ , (16) is reduced to

$$\begin{aligned}
H_1 &= \{ \text{Happens}(\text{Write}(\text{file}), t_m), \text{Happens}(\text{Read}(\text{file}), t_m) \} \\
H_2 &= \{ \text{Happens}(\text{Write}(\text{file}), t_m) \}
\end{aligned} \tag{17}$$

Written as a disjunction

$$\begin{aligned}
&(\text{Happens}(\text{Write}(\text{file}), t_m) \wedge \text{Happens}(\text{Read}(\text{file}), t_m)) \\
&\vee \text{Happens}(\text{Write}(\text{file}), t_m)
\end{aligned}$$

Which is equivalent to

$$\text{Happens}(\text{Write}(\text{file}), t_m)$$

Thus the only hypothesis is,

$$H_1 = \{ \text{Happens}(\text{Write}(\text{file}), t_m) \}$$

The case of  $t_m < t_a$  must be investigated further. The resolution in Figure 6.1 shows that  $H_2$  is refuted if  $t_m < t_a$ . The resolution in Figure 6.2 shows that  $\text{HoldsAt}(\text{Modified}(\text{file}, c(t_m)), t_{\text{Obs}})$  in  $H_1$  is accepted for  $t_m < t_a$ . The resolution for  $\text{HoldsAt}(\text{Accessed}(\text{file}, c(t_a)), t_{\text{Obs}})$  would look exactly like the one shown in Figure 6.2 and is omitted here. From these resolutions, it can be concluded that only  $H_1$  is accepted.

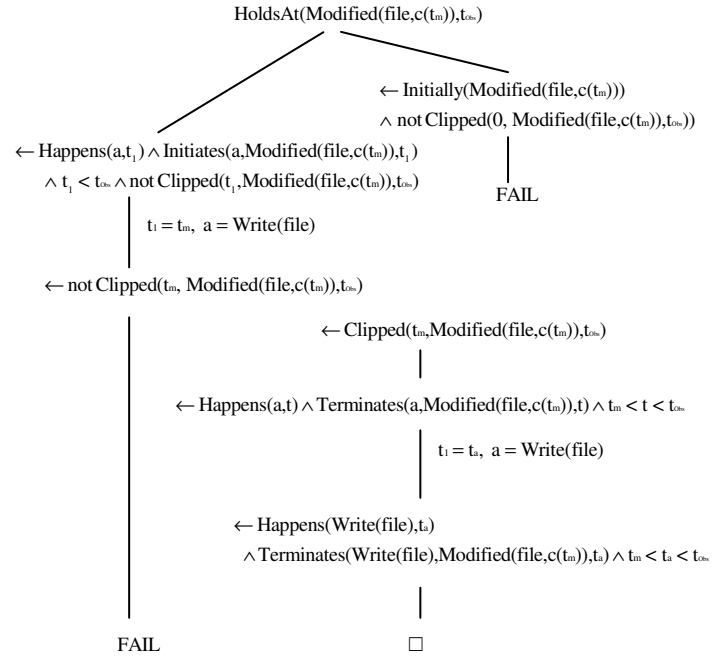


Figure 6.1 – Proposition fails for  $H_2$  when  $t_m < t_a$

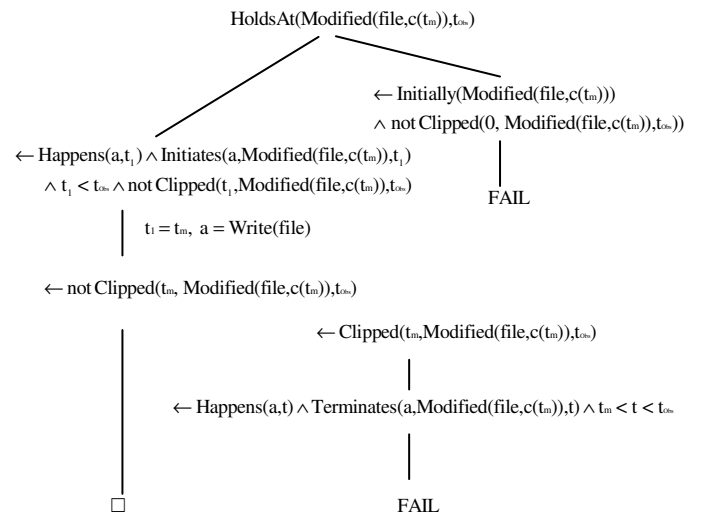


Figure 6.2 – Proposition does not fail for  $H_1$  when  $t_m < t_a$

In the case of  $t_a < t_m$ ,  $H_1$  is refuted, as shown in Figure 6.3. A resolution for  $H_2$  would look exactly like the resolution in Figure 6.3, with  $\text{Accessed}$  and  $\text{Read}$  replaced with  $\text{Modified}$  and  $\text{Write}$ .

Thus,  $H_2$  is also refuted for  $t_a < t_m$ , showing that in the simple file system,  $t_a < t_m$  cannot occur.

It has then been shown that for observations of every file in the simple file system:

$$t_m \not> t_a$$

$$t_m = t_a \Rightarrow \text{Happens}(\text{Write}(\text{file}), t_m)$$

$$t_m < t_a \Rightarrow \text{Happens}(\text{Read}(\text{file}), t_a) \wedge \text{Happens}(\text{Write}(\text{file}), t_m)$$

These results would facilitate event reconstruction in a forensic investigation of the simple file system, since the sequence of events on can now be determined directly from the file timestamp configuration.

The requirements on timestamp evidence can be explicitly stated as:

$$t_m \leq t_a$$

This result is also interesting, since it impose restrictions on the formulation of a hypothesis for the system clock. Any occurrences of  $c(t_m) > c(t_a)$  in the observation set must necessarily mean that the clock has been adjusted backwards. If many files are available in the observation set, it can also be determined when the clock must have been adjusted.

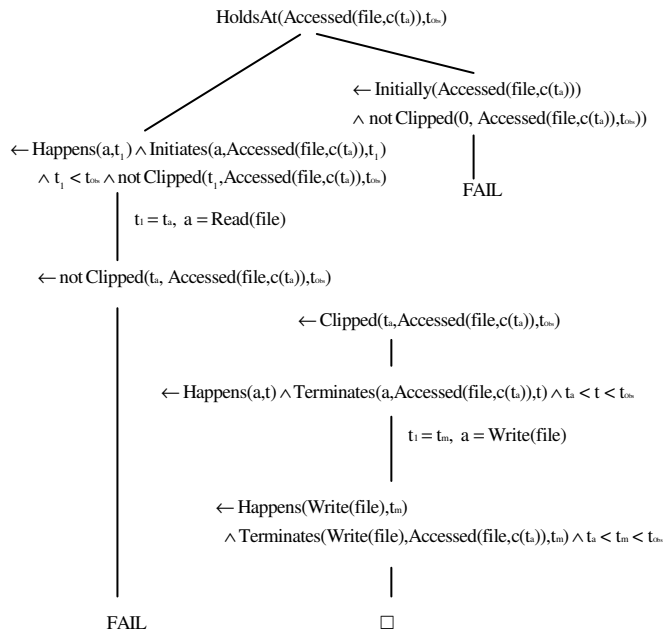


Figure 6.3 – Resolution of  $H_1$  when  $t_a < t_m$ .

## 7. RESULTS

In the previous sections, a simple file system model was defined in Simplified Event Calculus, and its properties were examined by SLDNF resolutions. When the set of actions A was known, the resolutions could be used to determine if a set of observations was consistent with the model. When the set of action A was unknown

but a set of observations O was known, the model was used to identify hypotheses of possible actions that could have occurred and test if these were consistent with the observation set O.

Simplified Event Calculus can be used to identify hypotheses of possible actions and test them for consistency with an observation set for an investigated system, as long as it is possible to determine how the system works (expressed by clauses in E) and the initial state (expressed by the clauses in S). This method can be used in digital investigations to identify hypotheses of which actions occurred on a system, and test them for consistency with the available evidence.

In Section 6, Simplified Event Calculus was used to derive invariants that must always hold for the simple file system. The method shown can be used for any system with known function and initial state to derive invariants. In digital investigation, these invariants can be used in tools that checks evidence from these specific systems for consistency. Specifically, in systems where timestamps are part of the model, the invariants can be used to determine adjustments of the system clock the timestamps are generated from.

## 8. CONCLUDING REMARKS

Simplified Event Calculus is a reasonable tool for building a model of a system and determining its properties in a hypothesis-based approach to digital investigation. By building such a model, the investigator can find and test possible hypotheses about actions that occurred on the system, and derive invariants for the system.

The approach in this work has been purely theoretical. Whether the Simplified Event Calculus can be used to model a real system under investigation, or if it is practical to do so, is an area of further study. In order to provide a full model of a real system in Simplified Event Calculus one must understand the system in full, something that can probably only be accomplished by studying the implementation details of the system. It might however be reasonable to construct a partial model only by studying the effects of operations on the real system, if it can be justified that the actions included in the model are the only actions of relevance in the investigation.

## 9. REFERENCES

- [1] B. Carrier, A hypothesis-based approach to digital forensic investigations. *CERIAS Tech Report 2006-06*, Purdue University, 2006
- [2] P. Gladyshev and A. Patel, Finite State Machine Approach to Digital Event Reconstruction, *Digital Investigation*, vol. 1, p. 19, 2004.
- [3] M. Shanahan, *Solving the frame problem : a mathematical investigation of the common sense law of inertia*. Cambridge, Mass.: MIT Press, 1997.