Modelling and Verification of Cash Withdrawal Transaction in Automated Teller Machine Using Timed Automata

Ari Ramadhana Hendrawan, Dieky Adzkiya
Department of Mathematics, Institut Teknologi Sepuluh Nopember, Kampus ITS
Sukolilo-Surabaya 60111, Indonesia
E-mail: ariramadhan01@gmail.com, dieky@matematika.its.ac.id

Abstract. This study was conducted to verify a system of Automated Teller Machine (ATM) which is a facility provided by a bank. Various transactions can be done by using ATM, including cash withdrawal, payment, and transfer. However, in spite of its function, ATM can also be a target of crime such as cash robbery and frauds. Therefore, the correctness of ATM and its security is essential and for that reason, formal verification is needed. Formal verification is a technique to ensure the model of a system to satisfy a certain specification. ATM has a time variable on its system. Therefore, timed automata can be used as a model of ATM. In this paper, an algorithm is constructed based on the ATM cash withdrawal steps. Next, we construct a timed automaton model and design nine specifications. Then, timed automaton were verified against these nine specifications using UPPAAL. From the verification results, it can be concluded that the security of the ATM system is guaranteed.

Keywords: Automated Teller Machine, Timed Automata, UPPAAL, Formal Verification.

1. Introduction
Automated Teller Machine (ATM) is one of the various facilities provided by a bank. Customers can do many transactions such as making cash withdrawals, transfers, making payments, and checking account balances. Besides those advantages, ATM can be the target of crimes, such as robberies [1]. Therefore, the ATM system must be guaranteed so that its security is also guaranteed. There is a method in the literature to ensure that the ATM system is running correctly by using simulation. However, there is a drawback that the assumptions given for the system may not be realistic. Hence, formal verification is chosen to ensure the correctness and safety of the system. It can mathematically ensuring the model of a system satisfies some specifications. The system is confirmed to be correct if the system satisfies all the properties obtained from the specifications [2]. Many research that discusses verification on ATM asserted that it has passed the validity and security test by verification [1, 3, 4].

The current model of an ATM system is using untimed model. As such, some parts or activities that should take a certain amount of time have not been tested or modeled. Testing is a very precise step to ensure the correctness of a system. However, testing can never truly identify all errors in the system [1]. Therefore, in order for verification to be carried out on an
ATM system, it requires a model that has a time variable. Thus, in this paper, an ATM system is modeled using timed automata.

Timed automaton is a machine with a finite state extended by a time variable (clock) [5]. The time variable represents the time on the system which is initialized with zero when it is idle and increases with the same synchronization and ratio in real numbers. ATM is a real-time and safety critical system. As a real-time system, ATM has a high level of complexity because it contains a complex network of processes in the form of data composition. Timed automata are a suitable model for the ATM system since several timed automata can be composed as a single timed automaton. There are many software for verification of timed automata. In this paper, we choose UPPAAL because it is user friendly. UPPAAL (Uppsala and Aalborg University) is a tool box for validation (through simulation of graphs composed of vertices and edges) and verification (automatic model checking) of a real time system [6].

This paper is written as follows. In the next section, we discuss some preliminaries needed in the subsequent sections, such as timed automata, simulation, and verification. Then, we construct the algorithm and its model based on timed automata. The specifications are designed based on TCTL (Timed Computation Tree Logic) and verification results are discussed in the next section. Finally, the conclusions are described in the last section.

2. Preliminaries

2.1. Timed Automata

A timed automaton is essentially a finite automaton (that is a graph containing a finite set of nodes or locations and a finite set of labeled edges) extended with real-valued variables [6]. The variables modeled as logical clocks in the system that are initialized with zero when it is started. Clock increases synchronously with the same rate. Clock constraints i.e. guards on edges are used to restrict the behavior of the automaton. A transition represented by an edge can be taken when the clocks’ values satisfy the guard labeled on the edge. Clocks may be reset to zero when a transition is taken. A timed automaton is a finite directed graph annotated with conditions over and resets of non-negative real valued clocks [5].

**Definition 1 (Timed Automata [6])** A timed automaton is defined as a 6-tuple $\mathcal{A} = (N, l_0, C, \Sigma, E, I)$ where

- $N$ is a set of locations (or nodes),
- $l_0 \in N$ is the initial location,
- $C$ is a set of clocks,
- $\Sigma$ is a set of actions,
- $E \subseteq N \times B(C) \times \Sigma \times 2^C \times N$ is the set of edges,
- $I : N \rightarrow B(C)$ assigns invariants to locations.

We shall write $l \xrightarrow{g,a,r} l'$ when $(l, g, a, r, l') \in E$.

The semantics of timed automaton is defined as a transition system where the state or configuration location contains the current location and value of clocks. There are two types of transitions between states. Automaton can be delayed for some time (a delay transition) or follow the visible side (action transition) [6]. Changes in clock values can be tracked using a function called clock assignments that map from $C$ to a non-negative real number $\mathbb{R}_+$. Let $u, v$ be denoted for a functions and $u, v \in g$ to denote the clock value denoted by $u$ satisfying guard $g$. Suppose $u + d$ for each $d \in \mathbb{R}_+$, denoted as a clock assignment where all $x \in C$ is mapped to $u(x) + d$ and to $r \subseteq C$ Also, let $[r \rightarrow 0]u$ be the clock assignment where all clocks in $r$ are mapped to 0 and corresponds to $u$ to the other clocks in $C \setminus r$ [6].
Definition 2 (Operation Semantics [6]) The semantics of an automaton is a transition system, it can also be called a timed transition system where each state is paired with \((l, u)\) and the transition is defined according to the following conditions:

- \((l, u) \xrightarrow{d} (l, u + d)\) if \(u \in I(l)\) and \((u + d) \in I(l)\) for non-negative real \(d \in \mathbb{R}_+\)
- \((l, u) \xrightarrow{r} (l', u')\) if \(l \xrightarrow{gr} l', u \in g, u' = [r \rightarrow 0]\) and \(u' \in I(l')\)

In general, timed automata are often composed in a network of timed automata as a set of clocks and actions, containing \(n\) timed automata, i.e. \(A_i = (N_i, l_i^0, C, \Sigma_i, E_i, I_i), 1 \leq i \leq n\).

Definition 3 (Semantics of a network of Timed Automata [5]) Let \(A_i = (N_i, l_i^0, C, \Sigma_i, E_i, I_i)\), a network of \(n\) automata. Let \(l_0 = (l_i^0, \ldots, l_n^0)\) be the initial vector. The semantics are defined as transition \((S, s_0, \rightarrow)\) where \(S = (N_1 \times \ldots \times N_n) \times \mathbb{N}^n\) be state with, \(s_0 = (l_0, u_0)\) is an initial state and \(\rightarrow \subseteq S \times S\) be a transition defined by:

- \((\bar{l}, u) \xrightarrow{d} (\bar{l}, u + d)\) if \(\forall d' : 0 \leq d' \leq d \Rightarrow u + d' \in I(\bar{l})\),
- \((\bar{l}, u) \xrightarrow{r} (\bar{l}'[l_i/l_i], u')\) if there exist \(l_i \xrightarrow{gr} l_i'\) s.t. \(u \in g, u' = [r \rightarrow 0]u\) and \(u' \in I(\bar{l})\),
- \((\bar{l}, u) \xrightarrow{r} (\bar{l}'[l_i/l_i, l_j/l_j], u')\) if there exist \(l_i \xrightarrow{c} l_i'\) and \(l_j \xrightarrow{e} l_j'\) such that \(u \in (g_i \land g_j), u' = [r_i \cup r_j \rightarrow 0]u\) and \(u' \in I(\bar{l}'[l_i/l_i, l_j/l_j])\).

![Figure 1](image)

**Figure 1.** A simple example of a timed automaton.

Figure 1(a) shows an automaton which is synchronized with the automaton in Figure 1(b) that has one location: loop. The Observer has two locations: idle and taken, where location taken is defined as committed location i.e. it has no time delay. If the Test is generated, the idle location on the Observer will move to the taken location according to the given synchronization: reset. Furthermore, taken can move to idle without any delay. Clock \(x \geq 2\) is used to start the transition after 2 units of time.

2.2. Reachability Analysis
Reachability analysis algorithm checks for the existence of a final location based on the initial target location [6]. Let \((l_0, D_0)\) defined as the initial state: \(l_0\) and \(D_0\) as the initial zone then the algorithm will search for the location containing \(l_f\) and the zone intersect with \(\varphi_{f_j}\). Then, the algorithm checks for transitions. Each transition is checked for a zone that intersects with \(\varphi_f\). If this condition is satisfied, the state is called reachable or fulfilled.

2.3. Formal Verification
In general, formal verification is a method to ensure that an object or system met its expectations or specifications. Basically, system verification techniques are based on a model that describes mathematical behavior and precision [2]. System verification is used to ensure a design or product under certain conditions meets its characteristics and properties. In other word,
Algorithm 1. Reachability Analysis

\[
\begin{align*}
\text{PASSED} &= \emptyset, \quad \text{WAIT} = \{(l_0, D_0)\} \\
\text{While} \quad \text{WAIT} \neq \emptyset & \\
\text{take} \quad (l, D) & \\
\text{if} \quad l = l_f \land D \cap \phi_f \neq \emptyset \text{ then return } \text{"YES"} & \\
\text{if} \quad D \not\subseteq D' \text{ for all } (l, D') \text{ PASSED then} & \\
\quad \text{add} \quad (l, D) \text{ to PASSED} & \\
\quad \text{for all } (l', D') \text{ such that } (l, D) \rightarrow k, G(l', D') \text{ do} & \quad \text{add} \quad (l, D) \text{ to WAIT} & \quad \text{end for} & \\
\text{end if} & \\
\text{end while} & \\
\text{return } \text{"NO"} & 
\end{align*}
\]

correctness of the system depends on the system specifications. The main purpose of model checking is to obtain system specification [5]. These specifications must be specific (formally well defined) and defined by using TCTL formula. The semantics of the TCTL formula are defined based on the tree that represents the automata as shown in Figure 2. The UPPAAL model checker is designed to check networks of timed automata by structuring them with the TCTL formula. By UPPAAL, the specifications can be structured with these following conditions:

a. A[φ] means in all state, φ always satisfied
b. E<>φ means there exist (at least one) state, φ eventually satisfied
c. A<>φ means in all state, φ eventually satisfied
d. E[φ] means there exist state, φ always satisfied

where φ are local properties which are checked in a state. In other words, a Boolean expression of the location, variable, and time constraint of guard and invariant. The illustration of the TCTL formula are shown in Figure 2 where the filled states are those for a given state formula φ holds. Bold edges are used to show the paths the formula evaluate on [5].

3. Modelling ATM System in Timed Automata

Before constructing the model, firstly we create the ATM system algorithm by using flowchart as shown in Figure 3. Furthermore, timed automata is constructed by using UPPAAL as a network of two automata: machine and user. We defined machine to represent ATM machine while user represents customers. First we determine the locations on the machine. The location on the machine is all the activities that the machine performs when a cash withdrawal transaction takes place, such as displaying a welcome screen, displaying the menu, withdrawing money, printing receipts, and releasing customer’s card after they are finished. After that, we construct the model by creating its locations. The locations are connected with the edges which represented the transition as shown in Figure 4.

The location is defined according to the activity that occurs, such as insertCardDisplay, languageOptions, validate, dispenseCash, menuOptions, and ejectCard by labelling the corresponding location. Next, we defined the clock variable y as guard and invariant. Guard and invariant are different: invariant can force the automata to change state when y is out of bounds while guard is not. Thus, invariant is given for all validation locations.

For user, we defined the locations according to customer’s activities such as: customers have the card but have not completed the transaction yet, not have the card and money, have the money but not the card, and have both of them. We defined it by labelling the locations with haveCardNotTheMoney, DontHaveCardAndMoney, haveMoneyButNotTheCard,
haveBothCardAndMoney as shown in Figure 5. Next, all of the synchronization on both automata defined in UPPAAL global declaration. Here we displayed the pseudo code of the declaration in Table 1.

### Table 1. Pseudo code for declaration of timed automata in UPPAAL

| Automata | Guard and Action |
|----------|------------------|
| machine = Template(); user = Template0(); | clock y; chan need; chan selectCancel, selectLanguage, chooseMenu; chan takeTheCard, takeTheMoney; chan enterNominal, inputRecentPIN, inputPIN, insertCard, inputNewPIN; |
| // List one or more processes to be composed into a system. system machine, user; |

4. **Properties and Verification Results**

In this section, we construct some properties or specifications that will be verified against the timed automata discussed in the previous section. Table 2 displays the nine specifications obtained based on the network of two timed automata that has been constructed. These specifications are used verify the automaton which is executed by UPPAAL by writing its corresponding TCTL formulas.

Units of time can be represented by $\Delta$ which equals to five seconds in this case study. Those nine specifications belong either to reachability or safety properties. The TCTL formula used in these properties are $E<>$ for reachability and $A[\ ]$ for safety. We will take the sixth specification into TCTL formula, according to machine it becomes $E<>$ cardLocked. UPPAAL can verify...
Figure 3. The flowchart of an ATM system.

this property automatically and also shown its transitions. One of the transitions and its result is shown below:

- (insertCardDisplay, haveCardButNotTheMoney)
- insertCard: user→machine
- (languageOptions, DontHaveCardAndMoney)
- selectLanguage: user→machine
- (inputPINDisplay, DontHaveCardAndMoney)
- selectCancel: user→machine
- machine
- (cardLocked, DontHaveCardAndMoney)

Figure 6 shows that UPPAAL has verified that E<> machine.cardLocked is satisfied by showing a green indicator. If it is red, it means the specification is not satisfied. For the rest of specifications, the result shows those are also satisfied. Hence, all of the specifications are reachable.
Figure 4. The timed automaton model of an ATM machine.

Figure 5. The timed automaton model of an ATM user.
### Table 2. A list of properties relevant to ATM systems

| Specifications |
|----------------|
| 1. If the machine dispenses cash, the amount of incorrectly entered PIN are less than 3 times. |
| 2. If the amount of incorrectly entered PIN equals 3, the customer's card is blocked automatically. |
| 3. If a customer entered cash nominal longer than 12∆, the machine displayed extra time. |
| 4. The machine validates card and PIN correctness not longer than 1∆. |
| 5. If a customer takes the card longer than 12∆, the card will be locked inside the machine. |
| 6. If a customer changes their PIN, the amount of incorrectly entered PIN must be less than 3 times. |
| 7. Locked card means deadlock. |
| 8. If a customer changes their PIN, the validation of their old PIN duration is not more than 1∆. |
| 9. If a customer changes their PIN, the validation of their new PIN duration is not more than 1∆. |

### Figure 6. One of verification results on reachability properties.

Safety properties are used to ensure that bad things will not happen such as bugs or errors. In general, a system should not find deadlocks. However, in ATM cash withdrawal flowchart, deadlocks are desired. So that we will tested that A[ ] deadlock will not satisfied. Based on the result before, it shows that E<>machine.cardLocked is satisfied which means that at least one
state satisfy its specification. Directly, it also means that $E<>\text{deadlock}$ is satisfied too. Hence, if we check all state always deadlock may not possible to reach. It is also proven by UPPAAL in Figure 7 below.

![Figure 7. One of verification results on safety properties.](image)

5. Conclusions and Future Work

The verification results state that the timed automata of the ATM system is suitable with the expectations (specifications) which required deadlock when the customer’s card is swallowed (situation when the customer does not take the card for more than one minute: $12 \Delta$). Based on the two properties of the verification reachability and safety properties, it can be concluded that the cash withdrawal flow with validation in the middle of the transaction is proven to be correct and safe i.e. there are no bugs and errors found. As a continuation of this work, we desired to verify the mobile banking and ATM over the same account held by two different people. Verification is used to check the balance deductions that occur when two people make transactions simultaneously such as money transfers and withdraw cash and both transfer money.

References

[1] (2018). ATM Logic Attacks. https://www.ptsecurity.com/ww-en/analytics/atm-vulnerabilities-2018/. accessed February 03 2020.

[2] Baier, C., Katoen, J, P, “Principles of Model Checking”, (2008), The MIT Press Massachusetts.

[3] Y, Wang., Y, Zhang., X, Li., H, Guo, ”The Formal Design Model of A Real-Time Operating System (RTOS+): Conceptual and Architectural Frameworks, International Journal of Software Science and Computational Intelligence 2”, (2010). 102–131.

[4] H, Shi., W, Ma., M, Yang., X, Zhang, “A Case Study of Model Checking Retail Banking System with SPIN, Journal of Computers 7”, (2012). 2503–2510.
[5] Behrmann, G., David, A., and Larsen, K. G, “A tutorial on UPPAAL 4.0”; (2008), Department of Computer Science, Aalborg University, Denmark.

[6] B. Johan., Y. Wang, “Timed Automata: Semantics, Algorithms, and Tools”; (2008), Lectures on Concurrency and Petri Nets. Uppsala University. 87-124.