Solving Equations over Automata with Timeouts

Ekaterina Shirokova
Tomsk State University
Tomsk, Russia

Nina Yevtushenko
Ivannikov Institute for System Programming of RAS
Higher School of Economics
Moscow, Russia

***Abstract*—In this paper, we consider the problem of solving equations for automata with timeouts over the concatenation operator. Such equations over finite automata are used for checking the vulnerability of web services to various attacks based on the analysis of the solvability of a corresponding equation. Since a number of web services contain time aspects in their descriptions, in this paper, we discuss the solvability problem for equations over automata with timeouts. We discuss two ways for solving these equations: an approach that is based on the selection of an appropriate submachine and an approach that is based on solving an equation over abstractions of automata with timeouts.**

Keywords—finite automata with timeouts, automata equations over concatenation operator

I. Introduction

The provision of various services using the Internet contributes to the emergence of a large number of web services, that is, software systems that provide interaction between the service provider and the client [1]. The client application sends a request to the server that satisfies the rules of the data transfer protocol, which is processed by the server. After processing the request, the server sends a response to the client, which is the requested information or an error message.

Web service implementations are usually cross-platform, since it does not matter where the service components are physically located and/or the language in which one or another module of the service system is written. Accordingly, service components can be implemented by different developers who typically do not share implementation details. In this regard, it is important to check both the functionality of service components and their security. Such a check makes sure that when using a third-party implementation, even if an incorrect request is sent, the system will not be "destroyed", and such a client will not be able to "get" to data that is not intended for it. The system should handle invalid input by generating appropriate messages.

In this work, attacks on a web service are understood as the implementation of actions aimed at unauthorized access to data, their modification, and so on. The non-profit OWASP [2] provides a list of the most popular attacks on web services, such as injections (for example, SQL injections); authentication failure; disclosure of confidential data; XSS attacks (cross-site scripting); incorrect security settings, etc.

Attacks on a web service can be staged by generating invalid input data. Such an implementation of attacks can be considered as a test for the robustness of the system under study: having received invalid data, the system must remain operational and must not compromise user data both inside the system and outside it. Detection of web service vulnerabilities can be done by solving automata equations. This approach allows to define a set of input sequences of impacts on a web service that can lead to an attack. Since a number of web services contain time aspects in their descriptions, in this paper, it would be interesting to discuss the solvability problem for equations over automata with timeouts.

Automata equations over the concatenation operation equations can come from the flow graph of the PHP program and attack patterns that are described via finite automata [3]. The intersection of the automaton that presents the set of possible actions of a web-service with the automaton that represents attack patterns allows drawing a conclusion whether some attacks can be successful. If such intersection is not empty, we can consider one of the following equations: a) $A.X \cong S$, b) $X.B \cong S$, c) $X.Y \cong S$. In these equations S corresponds to the intersection that was described above, so S is the automaton that corresponds to the set of non-safe actions of a web-service while a solution corresponds to possible malicious user inputs. Correspondingly, in cases a) and b) we solve the equation in order to describe such malicious user inputs; an obtained solution can be used in the sanitization. Case c) is more complex, in this case both prefix and suffix can depend of user inputs.

The problem of solving equations over the concatenation operator is discussed in [4, 5] where the formula is proposed for the largest solution of the language equation $A.X \cong S$. The authors argue that the equation $A.X \cong S$ has unique maximal solution, which includes all the other solutions to the equation (if equation is solvable). Similar results were obtained for an equation $X.B \cong S$.

In a number of cases, when a set of user inputs can lead to attacks, it becomes necessary to take into account time aspects. In this case, the problem of solving equations for automata with timeouts [6] over the concatenation operation arises. This problem is considered in this paper.

The structure of the paper is the following. Section II contains the necessary preliminaries: in Paragraph *A*, we remind the definition of finite automata and some properties and operations over the automata, while in Paragraph *B*, the definitions of an automaton with timeouts and its abstraction are considered. Section III is devoted to the investigation of the equations over $A.X \cong S$ and $X.B \cong S$ for automata with timeouts, and Section IV concludes the paper and contains discussions and the brief description of future work.

II. Basic definitions

A. A Finite Automaton

An *alphabet* [7] is a finite set of symbols (letters). The set of all finite strings over alphabet V is denoted by V^* and includes the empty string ε. A possibly infinite subset $L \subseteq V^*$ is a *language* over alphabet V.

A *Finite Automaton* or simply called an *automaton* throughout the paper is a quintuple $A = (A, V, \lambda_A, a_0, F_A)$ where A is a finite nonempty set of states with the initial state a_0 and the set F_A of final states, V is an alphabet, and $\lambda_A \subseteq A \times V \times A$ is a *transition relation*.

An automaton $\langle A', V, \lambda'_A, a_0, F'_A\rangle$ is a *submachine* of the automaton A if $A' \subseteq A$, $\lambda'_A \subseteq \lambda_A$, and $F'_A \subseteq F_A$. As usual, the transition relation λ_A of the automaton A is extended to sequences of actions over the alphabet V. A state a' is *reachable* from state a in A if there exists a sequence of consecutive transitions from state a to a'. An automaton is *connected* if each state $a \in A$ is reachable from the initial state.

Given a sequence α over alphabet V, α is accepted by A if α labels a sequence of transitions from the initial state to a final state. The *language* accepted by A, denoted L_A, is the set of all accepted sequences of A. Two automata A and B are *equivalent*, written $A \cong B$, if $L_A = L_B$.

An automaton A is *deterministic* if for each pair $(a, v) \in A \times V$ there exists at most one state $a' \in A$ such that $(a, v, a') \in \lambda_A$; otherwise, an automaton A is *nondeterministic*. In other words, an automaton A is *nondeterministic* if there exist $a \in A$ and $v \in V$ such that there are two different transitions $(a, v, a'), (a, v, a'') \in \lambda_A$ where $a' \neq a''$.

Given a nondeterministic automaton $A = (A, V, \lambda_A, a_0, F_A)$, there exists an equivalent deterministic automaton obtained from A by applying the so-called subset construction [7].

The *concatenation* of the automaton $A = (A, V, \lambda_A, a_0, F_A)$ with the automaton $B = (B, V, \lambda_B, b_0, F_B)$ is the automaton $A.B$ such that the language accepted by $A.B$ contains each string $\alpha\beta$ such that $\alpha \in L_A$ and $\beta \in L_B$. In this case, A is called the *prefix* of $A.B$, B is called the *suffix* of $A.B$. Formally [8], $A.B = (A \cup B, V, \lambda_S, a_0, F_S)$ where $F_S = F_B$ if $b_0 \notin F_B$ or $F_S = F_A \cup F_B$ if $b_0 \in F_B$. A transition relation λ_S is defined in the following way:

1. For each $s \in A \setminus F_A$, i.e., for each transition of the automaton A from a non-final state s, under action v we add a transition of A from state s under action v into the automaton $A.B$.
2. For each $s \in F_A$, i.e., for each transition of the automaton A from final state s under action v, and for a transition of the automaton B from initial state b_0 under action v we add two transitions into the automaton $A.B$: a transition of A from state s under action v and a transition of B from state b_0 under action v.
3. For each $b \in B$, i.e., for each transition of the automaton B from state b under action v we add a transition from state b under action v into the automaton $A.B$.

Thus, $A.B$ starts by stimulating A. When $A.B$ reaches a final state of A, $A.B$ starts stimulating B and this final state of A becomes the initial state of B. If the initial state of B is final then a final state of A is a final state of the concatenation. We also note that even in the case when A and B are deterministic automata the concatenation can still be nondeterministic according to the second rule of the concatenation construction.

Let L_1 and L_2 be languages on the alphabet V. The *right quotient* [5] of L_1 with L_2 is defined as $L_1/L_2 = \{\alpha : \alpha\beta \in L_1$ for some $\beta \in L_2\}$. The *left quotient* of L_1 with L_2 is defined as $L_2/L_1 = \{\beta : \alpha\beta \in L_1$ for some $\alpha \in L_2\}$.

B. An Automaton with Timeouts

In this paper, a *timed automaton* A is an automaton with timeouts, i.e. a 6-tuple $(A, a_0, F_A, J, \lambda_A, \delta_A)$, where A is a finite non-empty set of states with the initial state a_0, $F_A \subseteq A$ is a set of final states, J is an alphabet of (input) actions, $\lambda_A \subseteq A \times J \times A$ is a transition relation, $\delta_A: A \to A \times (N \cup \{\infty\})$ is a timeout function that determines the number of time units when the automaton can leave a current state without executing an action. If $\delta_A(a)\downarrow_N = \infty$, then the timed automaton stays at state a until an action is executed. The timeout at state a is denoted as T_a. In this paper, timed automata are considered in which all final states have infinite timeouts.

The timed automaton has an internal (clock) variable with non-negative integer values and indicates the time elapsed since the current state was reached. If a transition $(a, j, a') \in \lambda_A$, then the timed automaton A being in the state a changes its state to a' after executing the action j; the corresponding timed variable takes the value 0.

If for each pair $(a, j) \in A \times J$ there is at most one state $a' \in A$ such that $(a, j, a') \in \lambda_A$, then the timed automaton A is called *deterministic*. Otherwise, the timed automaton is called *non-deterministic*. We further consider only deterministic timed automata.

A timed (input) symbol (action) is a pair $(j, t) \in J \times Z_0^+$, where Z_0^+ is the set of non-negative integers. The timed symbol (j, t) indicates that the action j is executed at the time instance when the value of the timed variable is t. The sequence $\alpha = (j_1, t_1)(j_2, t_2)\ldots(j_n, t_n)$ where $t_1 \geq 0$ and $t_i \geq t_{(i-1)}$, $i = 2, \ldots, n$, of timed symbols is called a *timed* (*action*) *sequence* of length n. A timed action (j, t) is accepted by timed automaton A at state a_1 if j is a defined action at state a_1 and t is less than the timeout at this state. Let $t_0 = 0$. A timed sequence $\alpha = (j_1, t_1)(j_2, t_2)\ldots(j_n, t_n)$ is accepted by timed automaton A at state a_1 if there exists a sequence of states $a_2, \ldots, a_n$ such that $(a_k, j_k, a_{k+1}) \in \lambda_A$ and $(t_k - t_{k-1})$ is less than the timeout at state a_k, $k = 1, \ldots, n$. The set of all accepted timed sequences of A in the initial state a_0 is a *language* (*behavior*) L_A of timed automaton A.

The *concatenation* of the timed automaton $A = (A, a_0, F_A, J, \lambda_A, \delta_A)$ with the timed automaton $B = (B, b_0, F_B, J, \lambda_B, \delta_B)$ is the timed automaton $A.B$ such that the language accepted by $A.B$ contains each timed string $\alpha\beta$ such that $\alpha = (j_1, t_1)(j_2, t_2)\ldots(j_n, t_n) \in L_A$ and $\beta = (j_{n+1}, t_{n+1} - t_n)(j_{n+2}, t_{n+2} - t_n)\ldots\ldots(j_{n+k}, t_{n+k} - t_n) \in L_B$. Similar to classical automata, in this case, A is called the *prefix* of $A.B$, B is called the *suffix* of $A.B$.

The operations over automata with timeouts are not well defined and in order to formally define the concatenation of automata with timeouts we adapt the notion of an abstraction that was proposed for finite state machines with timeouts [6, 9] and will use this abstraction when solving equations over such automata. In this paper, it is proposed to construct a $\hat{I}$-automaton that in some cases, corresponds to the behavior of a timed automaton. Such an automaton has transitions over abstract action $\hat{I}$ that correspond to elapsing one time unit [7].

Consider a deterministic timed automaton $A = (A, a_0, F_A, J, \lambda_A, \delta_A)$. The corresponding $\hat{I}$-automaton is $A\hat{i} = (A\hat{i}, J\hat{i}, \lambda_{A\hat{i}}, (a_0, 0), F_A)$ where $J\hat{i} = J\hat{i} \cup \{\hat{I}\}$ while the set of states has a state (a, t) where t is a non-negative integer if there is a finite timeout T_a at state a that is bigger than t. If the timeout T_a at state a is the infinity, then there is state $(a, 0)$ in the automaton $A\hat{i}$. If $\delta_A(a) = (a', T_a)$ and the timeout $T_a < \infty$, then timeout transition from state a to state a' is transformed to a sequence

$((a, 0), \hat{I}, (a, 1))$ $((a, 1), \hat{I}, (a, 2))\ldots((a, T_a - 1), \hat{I}, (a', 0))$ of transitions, where all intermediate states (a, t), $t = 1, \ldots, T_a - 1$, are not final and «copies» of the state a, have the same transitions under $j \in J$ as the state a.

The set of all accepted sequences of $A\hat{\imath}$ in the initial state $(a_0, 0)$ is a *language* (*behavior*) $L_{A\hat{\imath}}$ of $\hat{I}$-automaton is $A\hat{\imath}$.

Proposition 2.1 Given a timed automaton A, a corresponding $\hat{I}$-automaton $A\hat{\imath}$ accepts a sequence $\hat{I}^{t_1} j_1 \hat{I}^{t_2} j_2 \ldots \hat{I}^{t_n} j_n$ at state $(a, 0)$ if and only if timed automaton A accepts a sequence $(j_1, t_1)(j_2, t_2 - t_1)\ldots(j_n, t_n - t_{n-1})$ at state a.

Consider an example. A timed automaton A and its corresponding $\hat{I}$-automaton $A\hat{\imath}$ are shown in Fig. 1(a) and Fig. 1(b). The timed automaton A in states a_0 and a_1 can stay infinitely long until an action is executed. In $\hat{I}$-automaton $A\hat{\imath}$, these states correspond to the states $(a_0, 0)$ and $(a_1, 0)$. In the state a_2 , a timeout transition to state a_0 is defined. Thus, if the action j_2 is not executed by timed automaton A in state a_2 within three time cycles, then A will move to state a_0. The timeout transition from state a_2 to state a_0 corresponds to the chain of transitions $((a_2, 0), \hat{I}, (a_2, 1))$ $((a_2, 1), \hat{I}, (a_2, 2))$ $((a_2, 2), \hat{I}, (a_0, 0))$ in the $\hat{I}$-automaton $A\hat{\imath}$. States $(a_2, 1)$ and $(a_2, 2)$ are «copies» of the state a_2, that is there are the same transitions under j_2 as at the state $(a_2, 0)$.

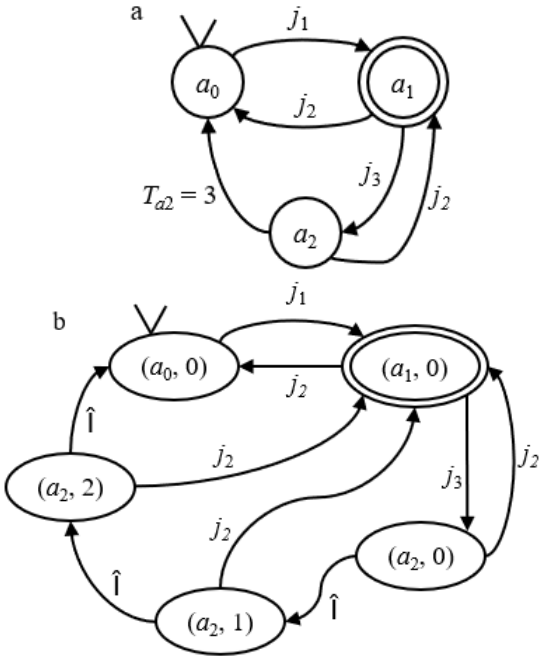

Fig. 1. (a) Timed automaton A; (b) Corresponding $\hat{I}$-automaton $A\hat{\imath}$

The *concatenation* of the $\hat{I}$-automaton $A\hat{\imath} = (A\hat{\imath}, J\hat{\imath}, \lambda_{A\hat{\imath}}, (a_0, 0), F_A)$ with the $\hat{I}$-automaton $B\hat{\imath} = (B\hat{\imath}, J\hat{\imath}, \lambda_{B\hat{\imath}}, (b_0, 0), F_B)$ is the automaton $A\hat{\imath}.B\hat{\imath}$ such that the language accepted by $A\hat{\imath}.B\hat{\imath}$ contains each string $\alpha\beta$ such that $\alpha = \hat{I}^{t_1} j_1 \hat{I}^{t_2} j_2 \ldots \ldots \hat{I}^{t_n} j_n \in L_A$ and $\beta = \hat{I}^{t_{n+1}-t_n} j_{n+1} \hat{I}^{t_{n+2}-t_n} j_{n+2} \ldots \hat{I}^{t_{n+k}-t_n} j_{n+k} \in L_B$. Similar to classical automata, in this case, $A\hat{\imath}$ is called the *prefix* of $A\hat{\imath}.B\hat{\imath}$, $B\hat{\imath}$ is called the *suffix* of $A\hat{\imath}.B\hat{\imath}$.

Proposition 2.2 Given timed automata A and B. $(A.B)\hat{\imath} \cong A\hat{\imath}.B\hat{\imath}$, the $\hat{I}$-abstraction of the concatenation A and B is the concatenation of corresponding $\hat{I}$-automata $A\hat{\imath}$ and $B\hat{\imath}$.

III. Solving Timed Automata Equations

In this paper, we consider two types of automata equations over the concatenation operator: $A.X \cong S$ and $X.B \cong S$ where the timed automaton X is the unknown. We can also consider the corresponding inequalities $X.B \leq S$ and $A.X \leq S$.

The timed automaton $Largest$ is called the *largest solution* to an equation $A.X \cong S$ ($X.B \cong S$) if it contains every solution of the equation.

We discuss two ways for solving these equations.

A. Solving an equation by the use of submachines

An algorithm for solving the equation $A.X \cong S$, has the following steps.

1. For the timed automaton S with the initial state s_0 we check whether there is a submachine A' with the initial state s_0 that is equivalent to A.
2. If such submachine A' exists, then it is necessary to add to the largest solution $Largest$ each submachine of the timed automaton S with the initial state a, where a is the final state of the submachine A'.
3. We construct the concatenation of the timed automata A and $Largest$. If the concatenation is equivalent to S then $Largest$ is the largest solution to the equation $A.X \cong S$; otherwise, the equation is unsolvable.

Proposition 3.1 The automaton with timeouts obtained in Step 2, is the largest solution to the inequality $A.X \leq S$.

The procedure for constructing a largest solution to the equation $X.B \cong S$ is a bit different from the above. For initial state s_0 of the timed automaton S, it is necessary to check whether there is a submachine X' with an initial state s_0, such that $X'.B \cong S$. Each such submachine X' (if it exists) is added to the largest solution $Largest$. Finally, it is necessary to assure that the timed automata $Largest.B$ and S are equivalent.

B. Solving an equation using the abstractions

Another way for solving the equation $A.X \cong S$ ($B.X \cong S$) is to use the corresponding abstractions for which the left (right) quotient operation [4, 5] is well developed.

Proposition 3.2 Timed automaton X' is a solution to the equation $A.X \cong S$ if and only if $\hat{I}$-automaton $X'\hat{\imath}$ is a solution to the equation $A\hat{\imath}.X\hat{\imath} \cong S\hat{\imath}$.

Proposition 3.3 Timed automaton X' is a solution to the equation $B.X \cong S$ if and only if $\hat{I}$-automaton $X'\hat{\imath}$ is a solution to the equation $B\hat{\imath}.X\hat{\imath} \cong S\hat{\imath}$.

Consider an example of finding a solution to a timed automata equation $A.X \cong S$. The timed automaton A and the corresponding $\hat{I}$-automaton $A\hat{\imath}$ are shown in Fig. 1. The timed automaton S and the corresponding $\hat{I}$-automaton $S\hat{\imath}$ are shown in Fig. 2.

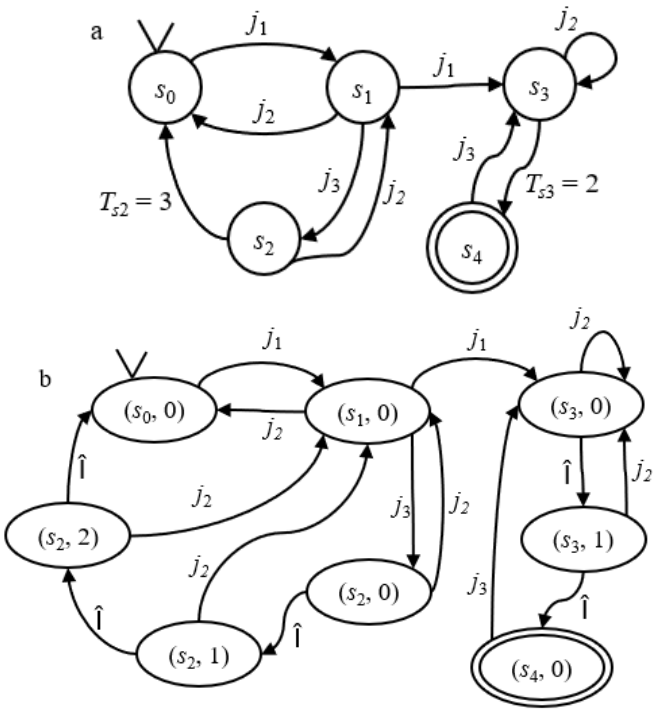

Fig. 2. (a) Timed automaton S; (b) Corresponding Î-automaton Sî

According to Proposition 3.2, in order to solve the equation $A.X \cong S$, we determine a solution to the equation Aî$.X$î $\cong S$î. An algorithm for solving the equation Aî$.X$î $\cong S$î, has the following steps or the left quotient can be utilized [5].

Consider the equation Aî$.X$î $\cong S$î. It can be seen that the Î-automaton X'î (Fig. 3(a)) is a submachine of the Î-automaton Sî and the concatenation of the Î-automaton Aî and X'î is equivalent to the Î-automaton Sî. Thus X'î is the solution of the Î-automata equation Aî$.X$î $\cong S$î. Next, we transform Î-automaton X'î into the corresponding timed automaton X', which is the solution of the equation $A.X \cong S$. States (s_1, 0) and (s_4, 0) of Î-automaton X'î correspond to states s_1 and s_4 in the timed automaton X'. States (s_3, 0) and (s_3, 1) of Î-automaton X'î correspond to the state s_3 in the timed automaton X'. The sequence of transitions ((s_3, 0), Î, (s_3, 1)) ((s_3, 1), Î, (s_4, 0)) corresponds timeout transition $Ts_3 = 2$ from state s_3 to state s_4. Thus, we get timed automaton X' (Fig. 3(b)). The concatenation of the timed automaton A and the obtained timed automaton X' is equivalent to the timed automaton S. Thus, X' is the solution to the equation $A.X \cong S$.

It should be noted that the solution of the equation $A.X \cong S$ ($X.B \cong S$) describes the malicious input of the web service and can be used to sanitize input data from malicious content. Any submachine of the largest solution to the equation $A.X \cong S$ ($X.B \cong S$) contains sequences whose concatenation with sequences of the automaton A (the automaton B) is unsafe.

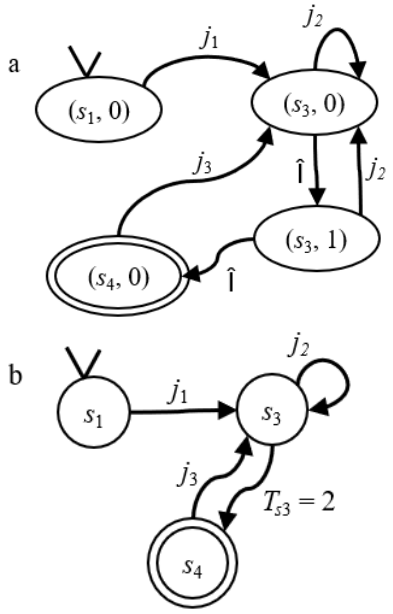

Fig. 3. (a) Î-automaton X'î; (b) Corresponding timed automaton X'

IV. Discussion and Conclusion

In this paper, we considered the problem of solving timed automata equations over the concatenation operator. Such equations appear when checking the security of a web service, which is a hot topic. An approach for solving such equations based on the selection of appropriate submachines is proposed. However, this approach requires further research which is a part of our future work. An approach that is based on solving an equation over abstractions also needs additional research, especially would be interesting to evaluate the complexity of both approaches. Our future work also includes the application of the proposed approaches to checking the security of real web services.

References

[1] E. Al-Masri, and Q.H. Mahmoud, "Investigating web services on the world wide web", Proceedings of the 17th international conference on World Wide Web, 2008, pp. 795-804.

[2] OWASP Top Ten, url: https://owasp.org/www-project-top-ten/

[3] F. Yu, M. Alkhalaf, and T. Bultan, "Generating vulnerability signatures for string ma-nipulating programs using automata-based forward and backward symbolic analyses," Technical Report 2009-11, UCSB CS, 2009.

[4] L. Kari, "On language equations with invertible operations," Theoret. Comput. Sci. 132 (1994), 129-150.

[5] P. Linz, An Introduction to Formal Languages and Automata, Jones & Bartlett Learning, 2011.

[6] O. Kondratyeva, Timed FSM strategy for optimizing web service compositions w.r.t. the quality and safety issues, Doctoral thesis, Paris Saclais, 2015.

[7] J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.

[8] A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Volume I: Parsing, Prentice Hall, 1972.

[9] O. Kondratyeva, N. Yevtushenko, and A. Cavalli, "Parallel composition of nondeterministic finite state machines with timeouts", Journal of Control and Computer Science of Tomsk State University, 2:27, 2014, pp. 73-81.