Kuwait J. Sci. **47 (1)** pp. 33-41, 2020

Specification and recognition of service trust behaviors

Hamdi Yahyaoui*, Jumana El-Qurna, Mohamed Almulla

Dept. of Computer Science, Kuwait University, Kuwait

**Corresponding author: hamdi@cs.ku.edu.kw*

Abstract

We propose in this paper a new approach for the specification and recognition of service trust behaviors through the so-called trust patterns, which denote typical trust behaviors. The trust patterns are defined based on context free languages and modeled as pushdown automata. Furthermore, we provide a unified model to capture these patterns and prove the correctness of the proposed model. Finally, we conduct a comprehensive experimental study using Weka in which we test different classifiers on a dataset of trust sequences and provide a comparative study regarding the accuracy of these classifiers.

Keywords: Data mining; pattern; pushdown automata; service; trust.

1. Introduction

Over the past few years, the development pace of services has been impressive. Academic and industrial efforts have been oriented towards improving the functional and non-functional aspects of services (Al-jazzaf, 2015). However, despite such tremendous efforts, several obstacles and challenges still restrict their widespread use in the development of large-scale, loosely-coupled, and trustworthy service oriented applications. One of these challenges is the lack of a common definition of service trust. Several researchers strived to provide a rigourous definition of trust. Olmedilla *et al*. (2005) consider that trust of an entity X to another entity Y regarding a service as the quantifiable belief of X in Y to act dependably during a specified period and within a specified context in relation to that service. Huang *et al*. (2010) perceive trust as a mental state comprising expectancy, belief, a trustee's competence and goodwill, and willingness to take risk. These informal definitions have helped to devise models for the computation of trust. However, there is still a lack of a formal definition of trust and consequently of service trust behaviors. Such a definition paves the way for performing sophisticated analysis such as classification, clustering, and verification of service behaviors. Accordingly, we propose in this paper a formal specification of service trust behaviors that is based on context-free languages.

To clarify, we consider a trust behavior as a sequence of observations, which are generated during a certain period of time. Sequences which denote typical behaviors are called trust patterns and are defined based on context-free languages. The rationale behind the adoption of such language style is to be able to define both regular and non-regular trust behaviors.

The devised framework would have a particularly added value on service selection because the services are classified into groups based on their trust behaviors. It is meant to guide users to deal with trustworthy services and detect the malicious ones. Furthermore, our framework could be applied to several fields such as e-commerce, e-auctions, e-services, etc.

The contributions of our work are three-fold:

- The formal specification of trust patterns from a language perspective. This specification captures regular and non-regular patterns.
- The elaboration of a deterministic pushdown automaton called SPIDER automaton that captures the aforementioned trust patterns.
- A set of experiments in which we test different classifiers on a synthetic dataset of trust sequences and provide a comparative study regarding the accuracy of these classifiers.

In Section 2, we present some research initiatives regarding modeling and computation of service trust. In Section 3, we devise context free languages to capture trust behaviors. We model the regular trust behaviors

using finite automata in Section 4. In Section 5, we model the SPIDER automaton that captures the proposed trust patterns. Finally, we provide in Section 6 experiments that show the adequacy of our modeling for data mining and in which we assess the accuracy of different classifiers on a synthetic dataset of trust sequences.

2. Related work

As the use of services dramatically increases, trust is becoming a major concern. Trust should exist between service users and service providers. A service provider is the entity that supplies the services. A service user is the entity that invokes a service. Service users need to know the service trust and which provided services meet their functional and nonfunctional requirements. Thus, trust is essential for providers and users to greatly improve service quality and selection.

Trust models can be classified according to the adopted modeling formalism: finite state machines (Carbone *et al*., 2003; Costa *et al*., 2011; Zhang *et al*., 2018), logic (Ashtiani & Azgomi, 2016; El-Qurna *et al*., 2017), probabilistic networks (Liu *et al*., 2018; Raj & Babu, 2017), petri nets (Bidgoly & Ladani, 2011; Guo *et al*., 2017), game theoretic models (Muchahari & Sinha, 2018; Hamed & Mohammad, 2017), hidden Markov models (Singh & Sinha, 2016; Yahyaoui & Zhioua, 2013). We review some of these prominent trust models.

Costa & Matteucci (2011) introduced the gate automata for specifying both security and trust policies. Gate automata can replace the enforcement mechanism that is controlling the behavior of an application. A major benefit of this approach is that it allows the optimization of the enforcement process based on trust levels. These automata work as edit automata for guaranteeing security properties and handling the trust levels of applications.

Carbone *et al*. (2003) proposed a formal model for the assessment of trust degree and trust transition in global computing-like environments. It leverages the past experiences of trusted certificate providers to assess the trust degree of trusted certificates. Trust is classified conceptually into six categories: disposition, situation, structure, belief, intention, and behavior. They use two ordering relations: trust ordering and information ordering as a solution for wrong decisions that result from confusing trust with knowledge.

Bidgoly & Ladani (2011) derived a new method for modeling and verification of reputation-based trust systems using Colored Petri Nets called TCPN. TCPN is an extension of Petri Nets, which combines the strengths of ordinary Petri nets with the strengths of a high-level programming language. The TCPN model is the first model that verifies the trust using a model-checking approach with temporal logic. It is independent from how to compute or represent the trust.

All the aforementioned research studies focused more on the computation of trust rather than the definition of the trust itself. Accordingly, the literature includes only informal definitions of trust and consequently trust behaviors. The formalization of trust behaviors is paramount for performing analyses such as logical verification, classification, and clustering. It is worth mentioning that our trust patterns definitions are based on our previous work (El-Qurna *et al*., 2017). We leverage here these definitions to build the aforementioned SPIDER model. We also explore data mining possibilities of the elaborated unified model rather than model checking trust behaviors as in (El-Qurna *et al*., 2017). The ultimate goal of this work is to derive highly accurate classifiers of trust behaviors.

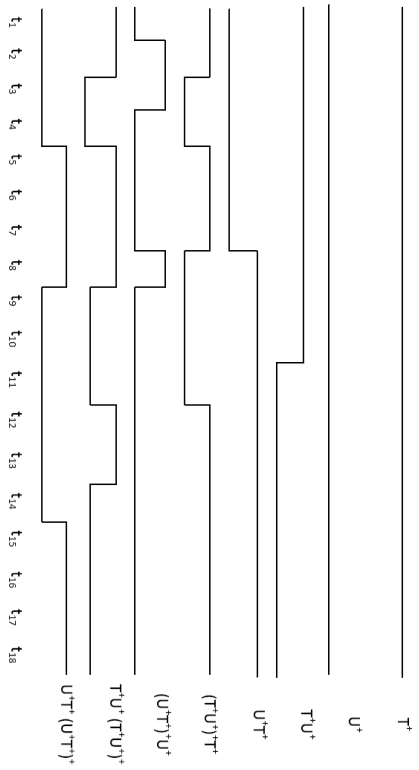


Fig. 1. Typical trust behaviors of services.

3. Trust patterns

In this section, we introduce our specification of service trust behaviors through the so-called trust patterns. We

adopt the informal definition of trust patterns that is provided in Yahyaoui and Zhioua (2013), which states that a trust pattern is a sequence of trust observations from which a clear conclusion about the behavior of web service can be drawn. Our idea relies on the ability to enumerate all possible trust observations sequences that capture both the regular and non-regular trust behaviors over $\{T,U\}^+$. As in Yahyaoui and Zhioua (2013), the sequence of observations mainly consists of the symbols T, U or both, where T denotes a Trustworthy observation while U denotes an Untrustworthy (malicious) observation.

We define the regular and the non-regular trust patterns, as:

- Regular trust patterns are regular languages that can be expressed using regular expressions over $\{T, U\}^+$ and can be recognized by finite automata.
- Non-regular trust patterns are context-free languages that can be expressed using context-free grammars over $\{T, U\}^+$ and can be recognized by pushdown automata.

Figure 1 shows all the sets of patterns that can be deduced over $\{T, U\}^+$. The set of sequences of one pattern can be encoded as a formal language which has properties that distinguish it. Based on Figure 1, we can conclude the following:

- The possible patterns are: T^+, U^+, T^+U^+, U^+T^+, $(T^+U^+)^+T^+$, $(U^+T^+)^+U^+$, $U^+T^+(U^+T^+)^+$, and $T^+U^+(T^+U^+)^+$.
- T^+ and U^+ patterns exhibit constant behaviors.
- T^+U^+ and U^+T^+ patterns alter their behaviors once during the execution time.
- $(T^+U^+)^+T^+$, $(U^+T^+)^+U^+$, $U^+T^+(U^+T^+)^+$ and $T^+U^+(T^+U^+)^+$ patterns alter their behaviors periodically with specific sequences at each time frame. These patterns represent the oscillating patterns from which we derive three regular and three non-regular patterns.

3.1 Regular patterns

In this section, we define the following regular service trust patterns.

- Trustworthy (T): The Trustworthy Language L_1 is $L_1 = \{T^n : n > 0\}$.
- Untrustworthy pattern (U): The Untrustworthy Language L_2 is $L_2 = \{U^n : n > 0\}$.
- Betraying (B): The Betraying Language L_3 is $L_3 = \{T^nU^m : m, n > 0\}$.
- Redemptive (R): The Redemptive Language L_4 is $L_4 = \{U^nT^m : m, n > 0\}$.
- Regular Oscillating pattern (RO): The Regular Oscillating language L_5 is $L_5 = (L_3 \,.\, L_3^* \,.\, L_1) \cup (L_4 \,.\, L_4^* \,.\, L_2)$.
- Oscillating to Betraying pattern (O-B): The Oscillating to Betraying language L_6 is $L_6 = L_3 \,.\, L_3 \,.\, L_3^*$.
- Oscillating to Redemptive pattern (O-R): The Oscillating to Redemptive language L_7 is $L_7 = L_4 \,.\, L_4 \,.\, L_4^*$.

3.2 Non-regular patterns

In this section, we provide three non-regular service trust patterns. The non-regular patterns are defined using the following theorem.

Theorem 1 (Hopcroft, 2007). If L_1 is a context-free language and L_2 is a regular language, then $L_1 \cap L_2$ is context-free language.

The two regular languages that will be used to define the Non-Regular patterns are:

- $L_{BT} = L_3 \,.\, L_3^* \,.\, L_1$.
- $L_{RU} = L_4 \,.\, L_4^* \,.\, L_2$.
- Non-Regular Oscillating pattern (NRO): The Non-Regular Oscillating language L_8 is $L_8 = \{w : n_U(w) = n_T(w)\} \cap (L_{BT} \cup L_{RU})$.
- Oscillating to Trustworthy pattern (O-T): The Oscillating to Trustworthy language L_9 is $L_9 = \{w : n_T(w) > n_U(w)\} \cap (L_{BT} \cup L_{RU})$.
- Oscillating to Untrustworthy pattern (O-U): The Oscillating to Untrustworthy language L_{10} is $L_{10} = \{w : n_U(w) > n_T(w)\} \cap (L_{BT} \cup L_{RU})$.

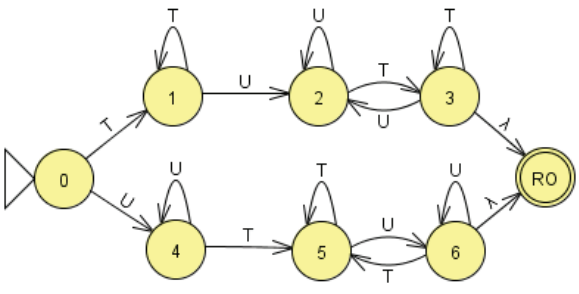


Fig. 2. NFA accepting the Regular Oscillating pattern

4. Finite automata for trust patterns

A finite automaton consists of states, control moves and inputs. The controls can be deterministic or non-deterministic. We choose the non-deterministic controls in our modeling to scale back the number of transitions and states. We model our regular trust patterns using NFA (Non-deterministic Finite Automata). These are the regular patterns: Trustworthy, Untrustworthy, Betraying, Redemptive, Regular Oscillating, Oscillating to Betraying and Oscillating to Redemptive.

Figure 2 outlines an NFA model for Regular Oscillating. We used the *JFLAP* tool (Gramond & Rodger, 1999) to model these patterns.

5. Pushdown automata for trust patterns

Pushdown automata are finite automata (Hopcroft, 2007) with additional capacity of a stack. There are two types of Pushdown Automata: Deterministic and Non-deterministic PDA. For accepting any input string utilizing PDA, there exist two models (Hopcroft, 2007): PDA that accept by final state and PDA that accept by empty stack. Our modeling uses DPDA which accept by final state. Our formal definition for the Pushdown automata follows the definition of Aho *et al*., 1968 by adding a right end marker to get a deterministic model. This reduces the complexity for each model and removes the guessing between transitions that may happen while moving.

To illustrate our non regular trust patterns formalization, we model the Non-Regular Oscillating pattern as a DPDA that is accepting by final state. Figure 3 outlines the DPDA for the Non-Regular Oscillating pattern.

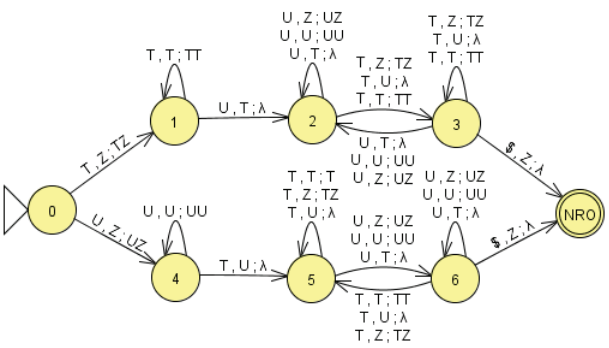


Fig. 3. A DPDA that accepts the Non-Regular Oscillating pattern

5.1 A unified SPIDER model

In this section, we introduce a unified model called "SPIDER automaton" that captures the aforementioned trust patterns. The main motivation to design the SPIDER automaton is to speed-up the recognition step instead of having one automaton per trust pattern.

The SPIDER automaton (DPDA that captures all the proposed trust patterns in this paper) is shown in Figure 4. The steps that should be followed to build the SPIDER automaton are detailed in (El-Qurna, 2016).

5.2 Correctness of the SPIDER automaton

The correctness of pushdown automata can be proved using either a mathematical induction or inductive assertion method of Floyd (Gopalakrishnan, 2006). Our correctness proof is carried out based on the mathematical induction. We proved the correctness of the SPIDER automaton but due to space limitations we omit the proof. The proof details can be found in El-Qurna, 2016.

6. Experimental results

Despite being well-defined, the space complexity of building a model that accepts only the input trust sequences. This is because it is exponential with the size of the input training sequences as proved in our previous work on trust sequence verification in El-Qurna *et al*., 2017. This creates challenges at the scalability level of our model. We demonstrate in this section that with less accuracy (but still high), data mining algorithms can classify these sequences even when the size is important. Of course, with few sequences, the ground truth can be generated with the SPIDER model. Our aim here is to achieve a good tradeoff between accuracy and complexity.

The objectives of the experiments are to:

1. Show that there are no overlaps between the trust patterns. This means that each sequence in the set of the input sequences belongs to one and only one trust pattern.
2. Test if our formal specification of trust patterns is good enough to be used to train classifiers to correctly label the trust sequences.
3. Establish a tradeoff between space complexity and accuracy by finding a good classifier for trust sequences and hence overcoming the space complexity issue that comes with the SPIDER model.

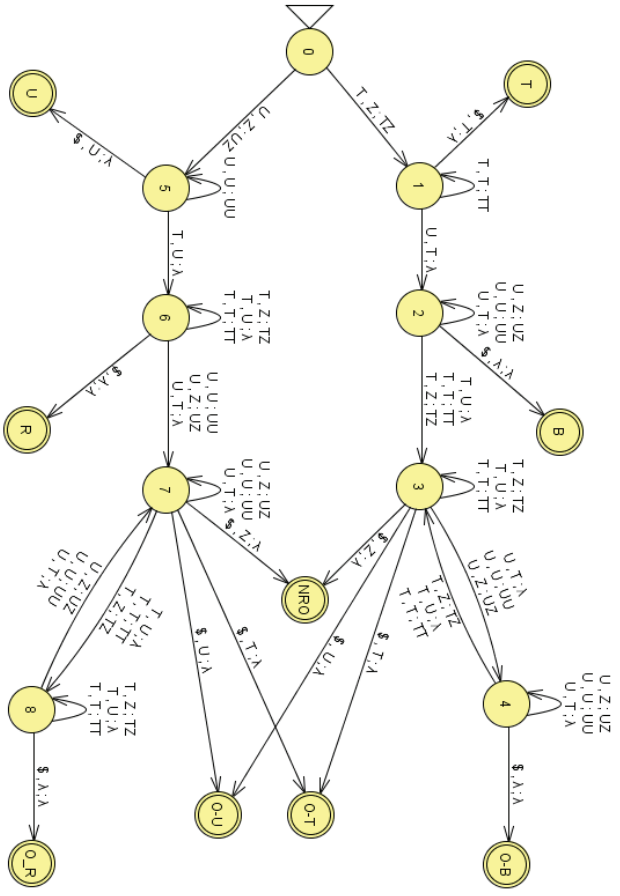


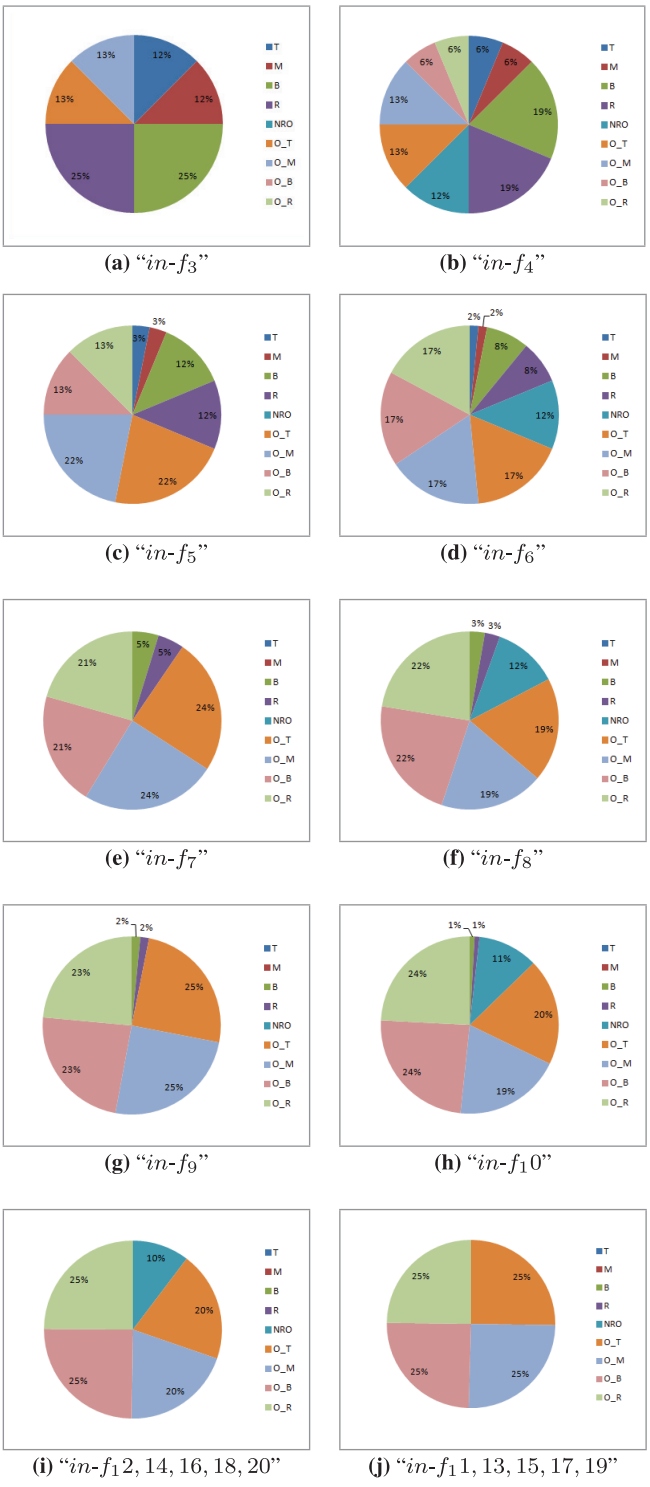
Fig. 4. SPIDER automaton

6.1 Dataset

In our experiments, we use a synthetic dataset that contains all the possible sequences over $\{T, U\}^+$. Our dataset consists of files called "in-f_i", where i denotes the length of sequences in the file. Each file contains 2^i sequences.

6.2 Simulating the SPIDER automaton

The SPIDER automaton was tested with our synthetic dataset using a C++ program that simulates it. A key characteristic of simulating the SPIDER automaton is to achieve the first objective that is mentioned above. To address this, we ensure that when the program terminates, it saves all the trust patterns that the sequence could reach. The program takes as input the files in-f_i, where $3 \leq i \leq 20$. Then it goes through each sequence in in-f_i deciding its patterns. The outputs of this simulation process are files called "out-f_i" one for each corresponding in-f_i. out-f_i contains the sequences of in-f_i, and each is followed by its trust pattern.



(a) "in-f_3" **(b)** "in-f_4"

(c) "in-f_5" **(d)** "in-f_6"

(e) "in-f_7" **(f)** "in-f_8"

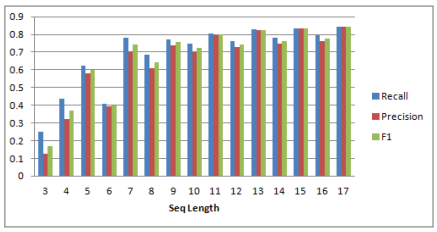
(g) "in-f_9" **(h)** "in-f_10"

(i) "in-$f_12, 14, 16, 18, 20$" **(j)** "in-$f_11, 13, 15, 17, 19$"

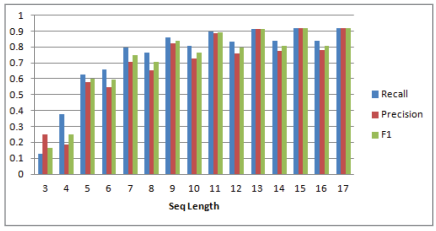
Fig. 5. The distribution results of trust patterns

6.3 Distributions of the trust patterns

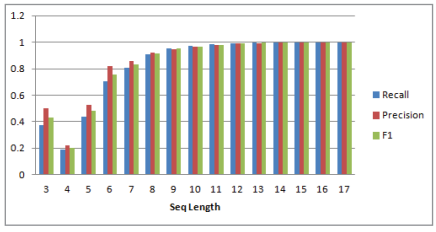
We investigate the distribution of the sequences for each trust pattern with respect to out-f_i, where $3 \leq i \leq 20$ (see Figure 5). It turns out that there are four dominant trust patterns. The dominant patterns are Oscillating to Trustworthy, Oscillating to Untrustworthy, Oscillating to Betraying and Oscillating to Redemptive.



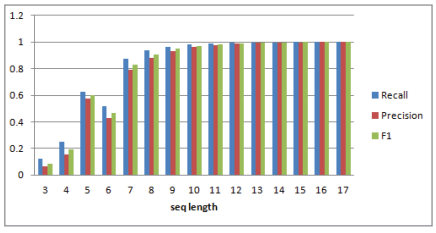
(a) J48 classifier



(b) Naive Bayes classifier



(c) Logistic classifier



(d) SMO classifier

Fig. 6. Recall, Precision and F1 of Weka classifiers.

Table 1. F1 values for J48, NaiveBayes, Logistic and SMO classifiers based on $out\text{-}f_i$: $3 \leq i \leq 17$.

Sequence length	Trees-J48	Bayes-NaiveBayes	Function-Logistic	Function-SMO
3	.17	.17	.43	.08
4	.37	.25	.20	.19
5	.60	.60	.48	.60
6	.40	.60	.76	.47
7	.74	.75	.83	.83
8	.64	.71	.91	.91
9	.76	.84	.95	.95
10	.72	.77	.97	.97
11	.80	.89	.98	.98
12	.74	.79	.99	.99
13	.83	.91	.99	1
14	.76	.80	1	1
15	.83	.92	1	1
16	.78	.81	1	1
17	.84	.92	1	1

Table 2. Accuracy results for J48, NaiveBayes, Logistic and SMO classifiers based on $out\text{-}f_i$: $3 \leq i \leq 17$.

Sequence length	Trees-J48	Bayes-NaiveBayes	Function-Logistic	Function-SMO
3	25%	12.50%	37.50%	12.50%
4	43.75%	37.50%	18.75%	25%
5	62.50%	62.50%	43.75%	62.50%
6	40.63%	65.63%	70.31%	51.56%
7	78.13%	79.69%	80.47%	87.50%
8	68.36%	76.56%	90.63%	93.75%
9	77.15%	85.74%	95.31%	96.48%
10	74.71%	80.76%	97.07%	98.05%
11	80.62%	89.75%	98.44%	98.93%
12	76.17%	83.30%	99.24%	99.41%
13	82.73%	91.21%	99.57%	99.68%
14	77.92%	83.85%	99.81%	99.83%
15	83.40%	91.91%	99.86%	99.91%
16	79.48%	84.05%	99.95%	99.95%
17	84.47%	91.98%	99.97%	99.97%

6.4 Data mining of trust sequences based on SPIDER automaton

In this part, our intent is to train the classifiers based on the files $out\text{-}f_i$: $3 \leq i \leq 17$. We use the Weka tool for the classification of trust sequences.

Three measures are important for classification: precision, recall and $F1$. $precision = \frac{tp}{tp+fp}$, where tp is the number of true positives, and fp is the number of false positive. $recall = \frac{tp}{tp+fn}$ where fn is the number of false negatives. Finally, $F1 = 2 \times \frac{precision \times recall}{percision+recall}$. The $F1$ values that we obtain from our classification process are provided in Table 1. We use a 10-fold cross-validation to assess the accuracy of the classifiers.

In Figure 6, we compare precision, recall and $F1$ values for J48, NaiveBayes and Functions-Classifier. The results show that J48 and NaiveBayes have irregular results unlike the Function-based classifiers which show good and stable results.

The Function-based classifiers in Weka give empirical evidence that our trust patterns are correctly specified. Table 2 shows the accuracy of all our experiments. Accuracy is measured by counting the proportion of correctly predicted examples in an unseen test dataset. From Table 2, we can conclude that:

- The results obtained for J48 and NaiveBayes are weak.
- The highest accuracy is 99.97% which is obtained using logistic-Classifier. Logistic-Classifier is a type of probabilistic statistical model that uses a multinominal regression.

7. Conclusion and future work

We presented in this paper a new approach for the specification of service trust behaviors. Typical trust behaviors are defined as patterns, which are specified using context-free languages. We also devised a common trust model called SPIDER, which is a deterministic push down automaton that captures all the trust patterns, and we proved its correctness. The experimental analysis shows that: (1) There are no overlaps between our trust pattern definitions, (2) The distribution of the patterns is balanced, and (3) The function-based classifiers provide good results based on SPIDER automaton specification.

Our work paves the way for sophisticated analysis such as model checking of service behaviors from a trust perspective, clustering and classification of services based on their trust behaviors, and forecasting service behaviors. As a future work, we plan to elaborate an algorithm that takes into consideration the probability in assessing the trust and which builds a probabilistic trust model for a service from trust observation sequences. Thereafter, we would formally verify if a probabilistic service trust model and satisfy specific trust properties. We also plan to extend our work to establish connections with Turing machines and include time properties to assess service trust behaviors.

References

Aho, A. and Corasick, M. (1975). Efficient string matching: An aid to bibliographic search. Communications of the ACM, **18**(6):333-340.

Aho, A., Hopcroft, J. and Ullman, J. (1968). Time and Tape Complexity of Pushdown Automaton Languages. Information and Control, **13**(3):186-206.

Aljazzaf, Z. (2015). Modelling and Measuring the Quality of Online Services. Kuwait Journal of Science, **42**(3).

Ashtiani, M. and Azgomi, M. (2016). Trust Modeling based on a Combination of Fuzzy Analytic Hierarchy Process and Fuzzy VIKOR. Computing, **20**(1):399-421.

Bidgoly, A. and Ladani, B. (2011). Trust Modeling and Verification Using Colored Petri Nets. In Proceedings of the eighth International ISC Conference on Information Security and Cryptology (ISCISC'2011), pages 1-8. IEEE Computer Society.

Carbone, M., Nielsen, M. and Sassone, V. (2003). A Formal Model for Trust in Dynamic Networks. In Proceedings of the International Conference on Software Engineering and Formal Methods (SEFM'2003), pages 54-61. IEEE Computer Society.

Costa, G. and Matteucci, I. (2011). Trust-driven Policy Enforcement Through Gate Automata. In Proceedings of the Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pages 208-215. IEEE.

El-Qurna, J., Yahyaoui, H. and Almulla, M. (2017). A New Framework for the Verification of Service Trust Behaviors. Knowledge-Based Systems, **121**:7-22.

Gopalakrishnan, G. (2006). Introduction to Automata Theory, Languages, and Computation Engineering: Applied Automata Theory and Logic. Springer.

Gramond, E. and Rodger, S. (1999). Using JFLAP to Interact with Theorems in Automata Theory. In Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education, pages 336-340.

Guo, J., Ma, J. and Wan, T. (2017). A Mutual Evaluation Based Trust Management Method for Wireless Sensor Networks. Chinese Journal of Electronics, **26**(2):407-415.

Hamed O. and Mohammad A. (2017). A Game-theoretic Approach to Model and Quantify the Security of Cyber-physical Systems. Computers in Industry, **88**:44–57.

Hopcroft, J. (2007). Introduction to Automata Theory, Languages, and Computation. Addison Wesley.

Huang, J. and Nicol, D. (2010). A Formal Semantics based Calculus of Trust. IEEE Internet Computing, **14**(5):38–46.

Jumana El-Qurna (2016). A model checking framework for the verification of service trust behaviors. Master's thesis, Computer Science Department, Kuwait University.

Liu, G., Liu, Y., Liu, A., Li, Z., Zheng, K., Wang, Y. and Zhou, X. (2018). Context-aware Trust Network Extraction in Large-Scale Trust-Oriented Social Networks. World Wide Web, **21**(3):713–738.

Muchahari, M. and Sinha, S. (2018). Game-Theoretic Method for Selection of Trustworthy Cloud Service Providers. In Proceedings of the First International Conference on Smart System, Innovations and Computing, pages 545-553. Springer.

Olmedilla, D., Rana, O., Matthews, B. and Nejdl, W. (2005). Security and Trust Issues in Semantic Grids. In

Semantic grid, volume 5271, pages 191-200.

Raj, E. and Babu, L. (2017). An Enhanced Trust Prediction Strategy for Online Social Networks using Probabilistic Reputation Features. Neurocomputing, **219**:412-421.

Singh, S. and Sinha, S. (2016). A Trust Model based on Markov Model Driven Gaussian Process Prediction. International Journal of Computer Applications, **146**(14).

Yahyaoui, H. and Zhioua, S. (2013). Bootstrapping Trust of Web Services based on Trust Patterns and Hidden Markov Models. Knowledge and Information Systems, **37**(2):389-416.

Zhang, T., Zheng, L., Wang, Y., Shen, Y., Xi, N., Ma, J. and Yong, J. (2018). Trustworthy Service Composition with Secure Data Transmission in Sensor Networks. World Wide Web, **21**(1):185-200.

Submitted : 23/04/2019

Revised : 28/05/2019

Accepted : 24/06/2019

التوصيف والتعرف على سلوكيات الثقة بالخدمة

حمدي يحياوي، جمانة القرنة، محمد الملا
قسم علوم الحاسوب، جامعة الكويت، الكويت

الملخص

نقترح في هذا البحث طريقة جديدة لتوصيف سلوكيات الثقة بالخدمة والتعرف عليها من خلال ما يسمى بأنماط الثقة، والتي ترمز إلى سلوكيات الثقة الاعتيادية. تم تعريف أنماط الثقة اعتماداً على اللغات التي لا تستند على السياق مع تعديلها لتصبح أوتوماتا منسدلة. علاوة على ذلك، نقدم نموذجاً موحداً لاحتواء هذه الأنماط وإثبات صحة النموذج المُقترح. وأخيراً وليس آخراً، نُجري دراسة تجريبية شاملة باستخدام Weka والتي نفحص بها عدة مصنفات تختبر مجموعة بيانات من متواليات الثقة كما نقدم دراسة نقارن فيها دقة هذه المصنفات.