Towards SOS Meta-Theory for Language-Based Security (Position Paper)

MohammadReza Mousavi
Department of Computer Science,
Eindhoven University of Technology

1 Introduction

SOS meta-theory [1] has been very successful in defining general criteria using which one can guarantee useful properties about the language constructs. These meta-theorems can save pages of standard proof thanks to their generic and language-independent formulation. Security properties of language constructs look like promising candidates to be turned into SOS meta-theorems and there has already been an attempt in this direction [8] in the context of process calculi security [2]. In this paper, we give an exploratory account of this issue in the context of language-based security [7]. A number of the ideas presented here can be taken directly to the process calculi security.

In the rest of this paper, we give a superficial overview of information-flow security [7] and in particular non-interference [3] as a central notion in this field. Then, we explore some interesting links between non-interference and our recent work on notions of bisimulation with data [4]. Some ideas regarding SOS meta-theorems for these notions will follow in Section 3. Section 4 concludes the paper and points out future work.

2 Non-Interference and Bisimulation

An important aspect of security is *confidentiality*. Confidentiality means that sensitive, or *higher-level*, information is never revealed in the course of interactions to *lower-level* users. In other words, confidentiality assures that higher-level information never leaks to lower-levels. A simplistic scenario for information leakage is through explicit assignment of high-level data items to low-level observable variables but it goes far beyond that. A low-level user may infer information about high-level data items by very implicit observations, exploiting so-called *covert channels*, e.g., by measuring execution time or power consumption.

Non-interference [3, 7] is an important means to assuring end-to-end confidentiality. It simply means that one cannot deduce anything about the high-level data/behavior by observing the low-level part of the system. In addition to confidentiality, non-interference has recently been exploited to support other aspects of security such as availability [9].

Suppose that we have a programming/specification language with two levels of confidentiality for data types. We denote the operational state of the program with $\langle p, h, l \rangle$ where p

is the program text, h is the higher level data and l is the low level data, all based on given domains P, H and L. Suppose that the operational semantics of a program is defined in terms of labelled transitions between the above-mentioned states with labels $\chi \in X$.

In the setting, a program is called non-interfering if *regardless of the higher-level data state*, it can always generate the same *behavior* as well as the lower-level data part during its execution. In order to formalize this informal explanation a number of choices has to be made. First of all a notion of behavior has to be fixed and here we choose the bisimulation semantics. Another important choice concerns the change in the higher-level data state. One may choose an open system semantics in which the higher-level data state can change arbitrarily during the execution or go for a closed system semantics in which higher-level data can only be changed by the entities specified in the system. We investigate both possibilities in the rest of this paper and propose two notions of non-interference, called *SL non-interference* and *ISL non-interference*, for open and closed systems, respectively.

Then, the following definitions (inspired by *low-bisimilarity* of [6] and bisimulation with data of [4]) are two possible formalizations of non-interference.

Definition 1 (SLNI Bisimulation and SL Non-Interference) A symmetric relation $R \subseteq P^2$ is called a *StateLess Non-Interference (SLNI) bisimulation relation* when $\forall_{(p,q)\in R}$, $\forall_{h_p,l,l',\chi,p',h'_p}$ $\langle p,h_p,l\rangle \xrightarrow{\chi} \langle p',h'_p,l'\rangle \Rightarrow \forall_{h_q}\ \exists_{q',h'_q}\ \langle q,h_q,l\rangle \xrightarrow{\chi} \langle q',h'_q,l'\rangle \wedge (p',q') \in R$. Programs p and q are *SLNI-bisimilar*, denoted by $p \underline{\leftrightarrow}_{slni} q$ when there exists an SLNI-bisimulation relation containing (p,q). A program p is *SL non-Interfering* when $p \underline{\leftrightarrow}_{slni} p$.

Note that unlike usual notions of bisimilarity, SLNI bisimilarity is not necessarily reflexive and hence, not an equivalence. Intuitively, the above non-interference definition requires for the non-interfering program to reproduce the same low-level data state regardless of the high-level state. The interesting part of the definition is that at each transition, the programs are compared using all possible high-level and all equal low-level data states. This resembles our notion of stateless bisimulation in [4]. As we motivate there, stateless bisimulation is very robust and compositional but it is usually very strong and difficult to establish. A similar observation can be made with respect to SLNI bisimulation and SL non-interference. An alternative for SL non-interference is the notion of ISL non-interference defined below.

Definition 2 (SBNI Bisimulation and ISL Non-Interference) A symmetric relation $R \subseteq (P \times H)^2$ is called a *StateBased Non-Interference (SBNI) bisimulation relation* when $\forall_{((p,h_p),(q,h_q))\in R}$, $\forall_{l,l',\chi,p',h'_p}$ $\langle p,h_p,l\rangle \xrightarrow{\chi} \langle p',h'_p,l'\rangle \Rightarrow \exists_{q',h'_q}\ \langle q,h_q,l\rangle \xrightarrow{\chi} \langle q',h'_q,l'\rangle \wedge ((p',h'_p),(q',h'_q)) \in R$. Programs p and q are *Initially StateLess Non-Interference (ISLNI)-bisimilar*, denoted by $p \underline{\leftrightarrow}_{islni} q$ when there exists an SBNI-bisimulation relation containing $((p,h_p),(q,h_q))$ for all $h_p,h_q \in H$. A program p is *ISL non-Interfering* when $p \underline{\leftrightarrow}_{islni} p$.

The above definition is motivated by the fact that low-level state can be observed and changed by low-level users while the change in the high-level state is in the hand of the system and if the system is closed, we need not cater for intermediate changes in the high-level states. Note that ISL non-interference is weaker that SL non-interference. We illustrate the above two definitions and their differences using the following simple examples.

Example 1 Consider a programming language with the terminating constant `skip`, the assignment, conditional (`if then else`) and the sequential composition (;) operators with the expected operational semantics. Assignment (:=) and condition (==) may compare and assign variables with/to values or other variables, respectively. Suppose that h is a high-level variable and l is a low-level one.

The following programs $l := h$ and `if` $(h == 5)$ `then` $l := 6$ `else` $l := 7$ are neither SL nor ISL non-interfering, for they lead to different behavior or low-level values depending on the initial value of the high-level variable h.

Also, `if` $(h == 5)$ `then` $h := 6$ `else skip` is neither SL nor ISL non-interfering since depending on the initial value of h, it immediately terminates or takes one more assignment step. This kind of behavior is a good source for a timing covert channel.

However, programs $h := 5$; $l := h$ and `if` $(h == 5)$`then` $h := 6$ `else skip` are both ISL but *not* SL non-interfering. In case there is no concurrent change to the higher-level variable, a low-level observer cannot infer anything about the higher-level variable by looking at different executions of the above programs. But by putting these programs in parallel with a higher-level component, we may observe different behavior and end-results depending on the intermediate values of the higher-level variable. For example, regarding the program $h := 5$; $l := h$, after execution of the first assignment the program evolves into $l := h$. It clearly does not hold that $l := h$ is non-interfering since the value of l is determined by, now not necessarily fixed, value of h.

3 On Rule Formats for Non-Interference

Structural Operational Semantics [5] is a commonly accepted method to define labelled transition semantics for languages. A semantic specification in the SOS style comprises a number of deduction rules defining possible transitions of a piece of syntax based on transitions of its constituting parts. Rule formats [1] define certain syntactic forms of deduction rules to be "safe" for certain purposes.

A distinguished class of rule formats is concerned with congruence of notions of behavioral equivalence. Translated into our terms, congruence of a behavioral equivalence usually means compositionality of the corresponding notion of non-interference. That is why in [8], a particular congruence format is used as a basis for a rule-format for proving non-interference. Following this approach, the `sfsl` and `sfisl` formats of [4] provide a convenient starting point. However, we intend to investigating the following possibilities for improving upon the format of [8] in our settings:

1. we would like to investigate separating the concerns of non-interference and its compositionality. This, in our mind, will simplify the resulting (this time, two separate) rule formats. Using one rule format one can check whether a non-interference property holds for a particular construct and using the other format one can check the robustness of the proven non-interference under different contexts. The rule format reported in [8] is, to our subjective judgment, too complicated to be understood and checked by a practitioner in this field and we hope that our proposal for separation of concerns will simplify the outcomes.

2. Secondly, we propose to study compositionality and non-interference for restricted language contexts and constructs, respectively. This is in contrast with the common practice of using SOS meta-theory for proving a property of a language as a whole. We can hardly imagine that any general-purpose language will provide compositional non-interference for all of its syntactically valid programs but rather, it is desirable to check whether a particular language construct (or a composed context) is non-interfering. For example, any language with a general assignment operator should not fit such a format while certain patterns of assignment can be easily proven to be non-interfering.

4 Conclusions

In this paper, we presented some ideas for notions of language-based non-interference based on notions of bisimulation with data. Subsequently, we suggested some starting points for devising a standard SOS format guaranteeing non-interference for restricted contexts. It still remains to research the initial ideas presented in this paper in order to propose a concrete format for language-based non-interference.

References

[1] L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, 2001.

[2] R. Focardi and R. Gorrieri. Classification of security properties (part I: Information flow). volume 2171 of *LNCS*, pages 331–396. Springer, 2001.

[3] J. A. Goguen and J. Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society, 1982.

[4] M. Mousavi, M. Reniers, and J. F. Groote. Congruence for SOS with data. In *Proceedings of LICS'04*, pages 302–313. IEEE Computer Society, 2004.

[5] G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Progamming*, 60:17–139, 2004.

[6] A. Sabelfeld. Confidentiality for multithreaded programs via bisimulation. volume 2890 of *LNCS*, pages 260–274. Springer, 2003, 2003.

[7] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

[8] S. Tini. Rule formats for compositional non-interference properties. *Journal of Logic and Algebraic Progamming*, 60:353–400, 2004.

[9] L. Zheng and A. C. Myers. End-to-end availability policies and noninterference. In *Proceedings of the CSFW'05*. To appear, 2005.