User Interface for Process Calculus for DoS-attack Resistance Analysis

Motoshi Ikeda and Shin-ya NISHIZAKI

Department of Computer Science, Tokyo Institute of Technology
2-12-1, Ookayama, Meguro, Tokyo 152-8552, Japan

nisizaki@163.com

*Corresponding author

Keywords: Process Calculus, User Interface, Interactive Reasoning.

Abstract. We proposed a process calculus for Denial-of-Service attack resistance, Spice calculus, which is an extended variation of Milner’s pi-calculus, adding a type system for cost estimation. Though we have studied theoretical aspects of the calculus, a practical application has not yet been developed. In this paper, we design and implement a user interface of reasoning in the Spice calculus. In the user interface, we harmonize both command-line and graphical interfaces. We explain the user interface, giving an example of an actual execution.

Introduction

In this section, we describe several background notions. The process calculus is a formal model of concurrent computation, where a computational step is formalized as a transition between expressions. The calculus of communicating systems (CCS) is a process calculus proposed by Milner [1] in 1980, which provides parallelism and synchronous communication. In 1992, Milner extended CCS as the pi-calculus [2][3][4] by adding name passing, name generation, and name restriction. The pi-calculus has been applied to various applications; the Spi-calculus is a variation of the pi-calculus [5] for

We proposed a new calculus for analysis of Denial-of-Service attack resistance, the Spice-calculus [6][7][8], which is an extension of the spi-calculus. We showed that we can describe the SYN-flood attack [6] and the signaling system [8]. We developed a cooperative technique of model checking and a network simulator using the spice calculus as the formal framework [7].

A theorem prover is software for checking proofs of formal logic. Many implementations of theorem prover have been developed during the last 50 years. A typical one from the early days is LCF (Logic for Computable Functions) [9] developed by Milner et al., in 1972. The successors of LCF are the HOL prover [10], Isabelle [11], and Coq [12]. A characteristic of the family of these provers is backward interactive reasoning [9] [10].

Other than these theorem provers, model checkers have become popular recently. A model checker is an automatic theorem prover, which verifies the correctness of a given temporal formula under a given automaton. A typical model checker is the SPIN model checker [13]. In the SPIN model checker, an input automaton can be described in the Promela modeling language. The various modeling languages are supported by model checkers. For example, the SLMC [14] model checker provides a pi-calculus for describing the input automaton.
Research Purpose

In this paper, we propose a new user interface for the reasoning of the process calculus. The main characteristic of the user interface is cooperation of the graphical representation and textual description, which creates substantial synergy. We implement the reasoning system of the Spice calculus and show concrete examples of the system.

User Interface for the Spice Calculus

The spice calculus [6] is a process calculus for analyzing Denial-of-Service attack resistance, which is an extension of the pi-calculus [4][2][3] and the spi-calculus [5].

A process is the primary expression of the spice calculus, which formulates a concurrent program, originally defined in the pi-calculus.

The definition of the processes is given in the previous paper [6]. The meaning of the processes is defined by the transition systems: reduction relation and commitment relation. The inner-process computation is formalized as the former and the inter-process computation as the latter. Computation costs are attached to each transition. The reduction relation is given as a ternary relation

\[ P > Q : c \]

among processes \( P \) and \( Q \) and a cost \( c \), which means that if a one-step computation from \( P \) to \( Q \) is carried out, then the cost \( c \) is entailed. The commitment relation is a quinary relation

\[ \mathcal{A} \vdash P \rightarrow^\alpha A : \sigma \]

among a configuration type \( \mathcal{A} \), processes \( P \), a name \( \alpha \), an agent process \( A \) and the cost assignment \( \sigma \). The agent process is an extension of the process including an intermediate form during communication. The configuration type conveys information on construction of parallel machines in which the process \( P \) is executed.

The two kinds of transition are defined inductively by rules. For example, the message receiving is defined by rules

\[
\text{CommIn} \quad a::\nu \vdash \text{inp } n (x) ; P \xrightarrow{\nu} \langle x \rangle P : \{a \cdot \text{store}_x\}
\]
User Interface for Reasoning of the Spice Calculus

In this work, we propose a good combination of textual and graphical interface of reasoning of the Spice calculus. Especially, we focus on the interactive inquiry of transition paths.

The overview of our reasoning system is presented in Fig. 1. In many areas of software application, the graphical user interface using a mouse and a touch screen is now popular and mainstream. However, as regards input, textual input is more effective and powerful than graphical input. Actually, the textual input interface is adopted by theorem provers such as the HOL prover [10], Isabelle [11], Coq [12], Mizar [15], etc. In these theorem provers, the output interfaces are textual. As seen in Fig. 1, the output interface becomes more efficient if it is both textual and graphical. In our reasoning system, we add the graphical output interface to the system. We show the harmonization of textual and graphical methods in the user interface by an actual example below. Our system is implemented in the programming language Java and executed on a MacBookAir with the following specifications: OS X El Capitan, Intel Core i7 (2.2GHz), and main memory 8GB.

![Figure 1. Overview of Reasoning System.](image)

We explain a simple example of process transition. We consider the process presented in Fig. 2.

```plaintext
1
2  
3     out C<A>; inp C(z); free z; stop
4  |
5     inp C(x); out C<x>; free x; stop
6  )
7  |
8     inp C<y>; out C<y>; free y; stop
9 )
```

![Figure 2. Process tarai.](image)

The process tarai is composed of three sub-processes, two of which communicate non-deterministically. If you apply the process to our system, it shows you output such as in Fig. 3.
In Fig. 3, a user chooses the verbose mode. The system shows that there are two options: one is communication between the first and the second processes, and the other is communication between the first and the third. Next, the user chooses the former option. Then, the system shows that there are two options: one is communication between the first and the second processes and the other is communication between the second and third.

In Fig. 4, the user chooses the simplified mode and then executes the expand command, which searches all possible transitions to depth 5 in the transition derivation tree.

During the execution in Fig. 4, our system generates the tree diagram in Fig. 5.
An attacker and a victim in the SYN-flood attack [6] is formalized as the process presented in Fig. 6. The tree diagram of the process is shown in Fig. 8.

Figure 6. Process of SYN-flood Attack.

Since it is more efficient to give textual inputs using a keyboard to the reasoning system than visual inputs using a mouse, our system and many theorem provers [12] [11] [15]. On the other hand, a textual output only in characters is often complicated to
read. Therefore, we display both textual and visual tree diagrams representing non-deterministic branches during execution of processes such as those in Figs. 5 and 8.

**Conclusion**

In this paper, we proposed a new design for a user interface of reasoning in the Spice calculus. In the user interface, both command-line and graphical interfaces are harmonized. We show the effectiveness of the user interface through the actual interaction between the user and the reasoning system. At the moment, our reasoning system supports the operational semantics formalized as a transition system. As a next step, we will apply our new design of the user interface to other forms of reasoning such as the analyzing system collaborated by a model checker and a network simulator [7]. There are many works [16] [17] on methods against attacks through the internet. It is also an interesting issue to study a new user interface for network defense and intrusion detection systems.

![Reduction/Commitment Start](image)

**Figure 7.** Execution of SYN-flood Attack Process in simplified mode.
Acknowledgement
This work was supported by Grants-in-Aid for Scientific Research (C) (24500009).

References

