Automatic analysis of firewalls using tree automata

Tony Bourdier

Automatic analysis of firewalls using tree automata

Tony Bourdier

INRIA Nancy Research Center – Pareo Team
615, rue du Jardin Botanique
54600 Villers-lès-Nancy, France
Tony.Bourdier@inria.fr

Abstract. Since the late 80s, firewalls are at the heart of network security. First designed to enable private networks to be opened up to the outside in a secure way, the growing complexity of organizations make them indispensable to control information flow within a company. The central role of firewalls in the security of the organization information make their management a critical task. That is why for years many works have been focused on checking and analysing firewalls. Nevertheless, most of these works are based on significant simplification of the firewall technology. Indeed, majority of algorithms developped for automatic analysis of firewalls are only founded on IP addresses filtering. This restriction does not allow to take advantage of these algorithms in real situations. In this paper, we propose a new approach for analysing firewalls, based on first order terms algebra and on tree automata techniques. Indeed, we attempt to specify using these techniques firewalls in a realistic way, that is to say taking into account packet filtering as well as the Network Address Translation (NAT) functionality. We next show that our framework allows us to compare firewalls and to perform automatic analysis of firewalls such as detection of classical misconfigurations.

1 Introduction

Since the late 1980s, firewalls are at the heart of network security. First designed to enable private networks to be opened up to the outside in a secure way, the growing complexity of organizations make them indispensable to control information flow within a company. The central role of firewalls in the security of the organization information make their management a critical task. That is why for years lots of works have been focused on checking and analysing firewalls. Nevertheless, most of these works are based on significant simplifications of the firewall technology. Indeed, the majority of algorithms developped for automatic analysis of firewalls is only founded on IP addresses filtering. For example, [CCBGA05,GL04,Liu08] specify firewalls as sets of rules of the form $R : s \in [n_1, n_2] \rightarrow \mathit{accept}/\mathit{drop}$. Others works [ASHBH05,UC04] assume that packets are not modified as they traverse the network, which is not the case in general. Indeed, it is common that firewalls separate a private network containing a lot of hosts which can access the internet using fewer IP than there are

in the private area. It is then necessary to map several private addresses into a single public one. That is why in spite of the good results obtained in current works, these restrictions do not allow to take advantage of developped algorithms in real situations. Moreover, some of the previous works are based on the fact that firewalls are essentially finite-state systems and perform their checking covering every path and every packet, which is not efficient in most of the cases. In this paper, we will attempt to show that terms algebra and tree automata techniques are very relevant to deal with problems on firewalls. Indeed, these techniques have already been shown suitable to perform cryptographic protocol verification but have not been explored for checking properties over firewalls. Thus, after having recalled required notions (Section 2), we propose in Section 3 a framework for specifying in a realistic way firewalls based on the representation of packets as first order terms and using tree automata for denoting sets of packets (since tree automata form a class having good properties), we next give a semantics to firewall specifications in our framework along with an algorithm to compare firewalls (Section 4) and show that analysis of firewalls such as the detection of misconfigurations is also performed in an automatic way in our framework (Section 5). At last, we will conclude by giving some perspectives and future possible works.

2 Preamble

This section briefly recalls basic notions used in this paper; more details can be found in [BN98,KK06] for first order terms and rewrite systems and in [CDG+08,Jac96] for tree automata and tree language theory.

A *signature* Σ consists in a finite set $\mathcal{S}_\Sigma$ whose elements are called *sorts*, possibly in relation by $<$ (which means "is a *subsort* of"), and an alphabet of symbols together with an application which associates to any symbol f a non empty sequence of sorts, which is denoted by $f : s_1 \times \ldots \times s_n \mapsto s$. $ar(f) = n$ is called *arity* of f. Given a signature Σ, a sort $\kappa \in \mathcal{S}_\Sigma$ and a countable set $\mathcal{X}^s$ of *variables* for each sort s, we denote by $\mathcal{T}^\kappa_{\Sigma,\mathcal{X}}$ the set whose elements are called *terms sorted by* κ inductively defined as follows: for any $x \in \mathcal{X}^\kappa$, x is in $\mathcal{T}^\kappa_{\Sigma,\mathcal{X}}$ and for any $f : s_1 \times \ldots \times s_n \mapsto \kappa$ and $\langle t_1, \ldots, t_n \rangle \in \mathcal{T}^{s'_1}_{\Sigma,\mathcal{X}} \times \ldots \times \mathcal{T}^{s'_n}_{\Sigma,\mathcal{X}}$, with $s'_i \leq s_i$ for any i, the word $f(t_1, \ldots, t_n)$ is in $\mathcal{T}^\kappa_{\Sigma,\mathcal{X}}$. $\mathcal{T}_{\Sigma,\mathcal{X}}$ is the union of $\mathcal{T}^s_{\Sigma,\mathcal{X}}$ for every sort s. Note that for any sorts κ and κ', either $\mathcal{T}^\kappa_{\Sigma,\mathcal{X}} \cap \mathcal{T}^{\kappa'}_{\Sigma,\mathcal{X}} = \varnothing$ or one of these sorts is a subsort of the other. The set of variables occuring in $t \in \mathcal{T}_{\Sigma,\mathcal{X}}$ is denoted by $\mathcal{V}ar(t)$. If any variable of $\mathcal{V}ar(t)$ occurs only once in t, t is said to be *linear*. If $\mathcal{V}ar(t)$ is empty, t is called a ground term. $\mathcal{T}_\Sigma$ denotes the set of all ground terms. A *position* of a term t is a finite sequence of positive integers describing the path from the root of t to the root of the sub-term at that position. The empty sequence representing the root position is denoted by ε. $\mathcal{P}os(t)$ is called the set of positions of t. $t_{|\omega}$, resp. $t(\omega)$, denotes the subterm of t, resp. the symbol of t, at position ω. We denote by $t\,[s]_\omega$ the term t with the subterm at position ω replaced by s. We call *substitution* any mapping from

$\mathcal{X}$ to $\mathcal{T}_{\Sigma,\mathcal{X}}$ which is the identity except over a finite set of variables $\mathcal{D}om(\sigma)$ called *domain* of σ extended to an endomorphism of $\mathcal{T}_{\Sigma,\mathcal{X}}$. σ is often denoted by $\{x \mapsto \sigma(x) \mid x \in \mathcal{D}om(\sigma)\}$. If $\mathcal{C}odom(\sigma) \subseteq \mathcal{T}_\Sigma$, σ is said to be ground. For any ground substitution σ, $\sigma(t)$ is called a *ground instanciation* of t. A *rewrite rule* (over Σ) is a pair $(lhs, rhs) \in \mathcal{T}_{\Sigma,\mathcal{X}} \times \mathcal{T}_{\Sigma,\mathcal{X}}$ such that $\mathcal{V}ar(lhs) \subseteq \mathcal{V}ar(rhs)$ and a *rewrite system* is a set of rewrite rules $\mathcal{R}$ inducing a *rewriting relation* over $\mathcal{T}_\Sigma$, denoted by $\rightarrow_\mathcal{R}$ and such that $t \rightarrow_\mathcal{R} t'$ iff there exist $(l, r) \in \mathcal{R}$, $\omega \in \mathcal{P}os(t)$ and a ground substitution σ such that $t_{|\omega} = \sigma(l)$ and $t' = t\,[\sigma(r)]_\omega$.

We call *n-ary (ascending) tree automaton* any quadruple $\mathcal{A} = \langle\, \Sigma, Q, F, \Delta \,\rangle$ such that Σ is a signature, Q is a finite set of *states*, F is a subset of Q whose elements are called *final states* and Δ is a relation over $\mathcal{T}_{\Sigma^n[Q]} \times Q$ whose elements are called *transitions* where Λ is a new symbol and $\Sigma^n\,[Q]$ consists of the unique sort $conf$ and the alphabet $(\Sigma \cup \{\Lambda\})^n \setminus \{\langle\, \Lambda, \ldots, \Lambda \,\rangle\} \cup Q$ such that $\langle\, f_1, \ldots, f_n \,\rangle \in (\Sigma \cup \{\Lambda\})^n$ is of sort $conf \times \ldots \times conf \mapsto conf$ with $ar(f_1, \ldots, f_n) = \max_{i\in[1,n]}(ar(f_i) \mid f_i \neq \Lambda)$ and $q : Q$ is of sort $\mapsto conf$. Examples of tree automata are given in **Appendix** A. An element of $\mathcal{T}_{\Sigma^n[Q]}$ is called a *configuration*. A transition $lhs \rightarrow rhs$ of Δ is *normalized* iff for any $\omega \neq \varepsilon$, $lhs(\omega) \in Q$. An automaton whose transitions are normalized is said normalized. A tree automaton is said *deterministic* iff all its transitions have a different left-hand side. Without loss of generality, we can consider that all automata are normalized and deterministic. The rewriting relation induced by Δ over $\mathcal{T}_{\Sigma^n[Q]}$ is denoted by $\rightarrow_\mathbb{A}$ and the language recognized by $\mathbb{A}$ is $\mathcal{L}(\mathbb{A}) = \{\langle\, t_1, \ldots, t_n \,\rangle \in (\mathcal{T}_\Sigma)^n \mid \exists q_f \in F, t_1 \otimes \ldots \otimes t_n \xrightarrow{*}_\mathbb{A} q_f\}$ where $t = t_1 \otimes \ldots \otimes t_n$ is the configuration such that: $\forall \omega \in \bigcup_{i=1}^n \mathcal{P}os(t_i)$, $t(\omega) = \langle\, t_1[\omega), \ldots, t_n[\omega) \,\rangle$ where $u[\omega) = u(\omega)$ if $\omega \in \mathcal{P}os(u)$ and Λ otherwise. A set E of n-tuples of terms (or equivalently n-ary relation) is said *recognizable* iff there exists an n-ary tree automaton $\mathbb{A}$ such that $E = \mathcal{L}(\mathbb{A})$. The following table recalls usual automata and operations over tree automata and their semantics:

Notation	Language recognized by the automaton
$\mathbb{A} \oplus \mathbb{A}'$	$\mathcal{L}(\mathbb{A}) \oplus \mathcal{L}(\mathbb{A}')$ where $\oplus$ is $\cap$, $\cup$, or $\times$
$\overline{\mathbb{A}}$	$(\mathcal{T}_\Sigma)^n \setminus \mathcal{L}(\mathbb{A})$
Ω^n_κ	n-tuples $\langle\, t_1, \ldots, t_n \,\rangle$ of $\mathcal{T}^\kappa_\Sigma$
Id^n_κ	n-tuples $\langle\, t, \ldots, t \,\rangle$ of $\mathcal{T}^\kappa_\Sigma$
$rec(t)$	ground instanciations of t (t linear)
$\sqcup_i(\mathbb{A})$	$(n+1)$-tuples $\langle\, t_1, \ldots, t_{i-1}, t, t_i, \ldots, t_n \,\rangle$ *s.t.* $\langle\, t_1, \ldots, t_n \,\rangle \in \mathcal{L}(\mathbb{A})$
$\sqcap_i(\mathbb{A})$	$(n-1)$-tuples $\langle\, t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n \,\rangle$ *s.t.* $\exists t \in \mathcal{T}_\Sigma : \langle\, t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n \,\rangle \in \mathcal{L}(\mathbb{A})$
$\sqcap_{i/t}(\mathbb{A})$	$(n-1)$-tuples $\langle\, t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n \,\rangle$ *s.t.* $\langle\, t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n \,\rangle \in \mathcal{L}(\mathbb{A})$

where n and i are integers, t is a term of $\mathcal{T}_{\Sigma,\mathcal{X}}$, κ is a sort and $\mathbb{A}$ and $\mathbb{A}'$ are n-ary tree automata. We also recall that the membership, the emptyness, the finiteness, the equivalence and the inclusion problem are decidable for tree automata.

3 Firewall specification

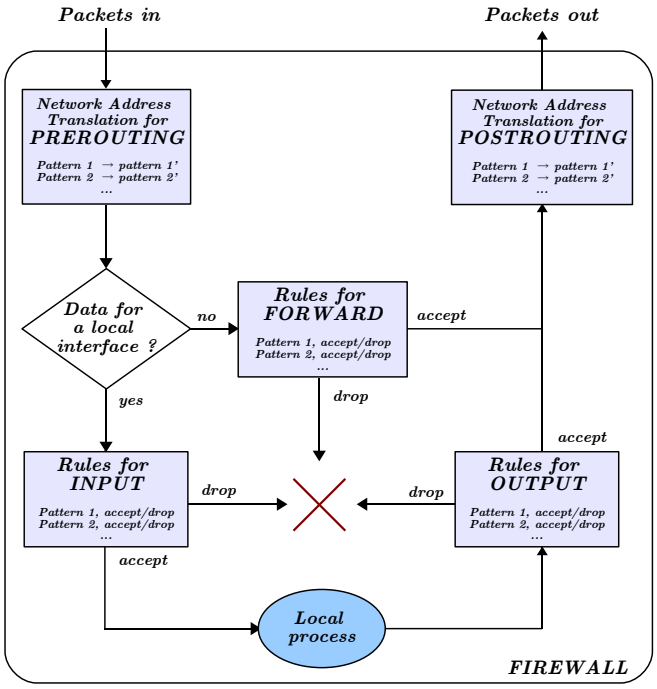
As said before, we try to consider in this paper firewalls not in a simplified way, that is to say not restricted to a transitional filter function.

3.1 Firewall background

Based on the description of firewalls embedded in Linux 2.4 [Rus02] and FreeBSD, NetBSD and other main operating systems [CF02], we define a firewall as an element of a network designed to control packets traffic between different domains by using a combination of:

- the packet filter technique, which consists in inspecting each packet and either allowing them to continue their traversal or dropping them;
- and the network address translation functionality, which consists in modifying network address information in packet headers.

The following scheme gives the global architecture of a firewall:



Thus, a firewall contains three filter blocks called *chains*: INPUT, OUTPUT and FORWARD. When a packet arrives to the entry of a filter block, the corresponding chain decides if the packet must be dropped or must continue its traversal of the diagram. In order to compute this decision, each chain is made up of a list of *rules* which map the description of a set of packets to a decision. The most often used criteria that chains use when inspecting packets are the following [CF02,Rus02]: the IP source address, the IP destination address, the protocol, the source port,

the destination port and the state of the session. Moreover, when a packet comes in or out, the firewall rewrites a part of the packet header. At the input of the firewall, we talk about *prerouting* and at the output we talk about *postrouting*. The former consists in modifying information concerning the destination of the packet and the latter information concerning the source of the packet.

3.2 Packet representation

First of all, we need to represent in a formal way packets we have to deal with. We propose to denote a packet as a first order term which will allow us to take advantage from powerfull existing techniques, namely tree automata. We thus define a signature $\Sigma_{Firewall}$ containing the contants 0 and 1 of sort Bit, TCP, UDP, ICMP of sort $Protocol$, new, $established$, $related$, $invalid$ of sort $State$ and the following non constant symbols:

$$\begin{array}{lll} bit & : & Bit \mapsto Binary \\ bin & : & Bit \times Binary \mapsto Binary \\ ip & : & Octet \times Octet \times Octet \times Octet \mapsto IP \\ port & : & Binary[16] \mapsto Port \\ packet & : & IP \times IP \times Protocol \times Port \times Port \times State \mapsto Packet \end{array}$$

where $Binary[n]$ is the subsort of $Binary$ containing terms built with $(n-1)$ symbols bin and $Octet$ is a shortcut for $Binary[8]$. Thus, the (real) octet $o = b_7.b_6.b_5.b_4.b_3.b_2.b_1.b_0$ is represented by the following term of sort $Octet$: $\overrightarrow{o} = bin(b_7, bin(b_6, bin(\ldots, bin(b_1, bit(b_0)))))$.

3.3 Packet filtering

As said before, packet filters consist in an ordered list of rules of the form $\langle\, set, decision \,\rangle$ where set is a set of packets and $decision$ is either ACCEPT or DROP. Usually, the sets of packets are represented by a tuple containing:

- an IP source (resp. destination) address interval $[ip_1^{src}, ip_2^{src}]$ (resp. $[ip_1^{dest}, ip_2^{dest}]$) which can be of the form 192.168.$*$.$*$ or under the form of an address together with a mask, *i.e.* either 192.168.1.8/29 or 192.168.1.8/255.255.255.248 (which denotes [192.168.1.8, 192.168.1.15]);
- a set pr of protocol names which is a subset of {TCP, UDP, ICMP} or the keyword any (to denote the set of all protocols);
- a range $[port_1^{src}, port_2^{src}]$ (resp. $[port_1^{dest}, port_2^{dest}]$) of numbers (or simply a number) between 1 and 65535, or the keyword any to denote the range $[1, 65535]$ for filtering the source (resp. destination) port;
- and a set of session states st which is a subset of {new, $established$, $related$, $invalid$} or any to denote the whole set.

In order to take advantage of tree automata techniques, we modelize sets occuring in filtering rules as tree automata. Thus, in order to simulate the behaviour of the filter part of the firewall, we have to build for any rule $\langle\, set, decision \,\rangle$ an automaton $rec(set)$ which recognizes the set of packets matching the rule. To

recognize intervals of IP and ports, we use the automaton denoted by $\mathbb{A}_{bin\_add}$ such that $\mathcal{L}(\mathbb{A}_{bin\_add}) = \{\langle\, b_1, b_2, b_1 + b_2 \,\rangle \mid b_1, b_2, b_3 : Binary[n]\}$ for some $n \in \mathbb{N}$ and whose construction is explained in **Appendix** B. From it, we build an automaton recognizing the pair of octets whose first element is less than the second one: $\mathbb{A}_{octet\_inf} = \sqcap_2(\mathbb{A}_{bin\_add} \cap \Omega^3_{octet})$ (in the same way we define $\mathbb{A}_{port\_inf} = \sqcap_2(\mathbb{A}_{bin\_add} \cap \Omega^3_{Binary[16]})$). We also define $\mathbb{A}_{ip\_inf} = \left\langle\, \Sigma_{firewall}, Q, \{q_F^{inf}\}, \Delta_{ip\_inf} \,\right\rangle$ the automaton recognizing the pair of IP $\langle\, ip_1, ip_2 \,\rangle$ such that $ip_1 \leq ip_2$ where Δ_{ip_inf} contains:

$$\langle\, ip, ip \,\rangle\, (q_F^{octet\_inf}, q_F^{\Omega^2_{octet}}, q_F^{\Omega^2_{octet}}, q_F^{\Omega^2_{octet}}) \to q_F^{inf}$$
$$\langle\, ip, ip \,\rangle\, (q_F^{Id^2_{octet}}, q_F^{octet\_inf}, q_F^{\Omega^2_{octet}}, q_F^{\Omega^2_{octet}}) \to q_F^{inf}$$
$$\langle\, ip, ip \,\rangle\, (q_F^{Id^2_{octet}}, q_F^{Id^2_{octet}}, q_F^{octet\_inf}, q_F^{\Omega^2_{octet}}) \to q_F^{inf}$$
$$\langle\, ip, ip \,\rangle\, (q_F^{Id^2_{octet}}, q_F^{Id^2_{octet}}, q_F^{Id^2_{octet}}, q_F^{octet\_inf}) \to q_F^{inf}$$

Q contains all states occuring in Δ_{ip_inf} and $q_F^{octet\_inf}$ is the final state of $\mathbb{A}_{octet\_inf}$. The range of IP in $[ip_1, ip_2]$ is recognized by $\mathbb{A}_{[ip_1,ip_2]} = \sqcap_{1/ip_1}(\mathbb{A}_{ip\_inf}) \cap \sqcap_{2/ip_2}(\mathbb{A}_{ip\_inf})$. Thus, we associate to any rule[1] $\langle\, set, decision \,\rangle$ the automaton $rec(set) = \langle\, \Sigma_{firewall}, Q_{set}, \{q_F^{set}\}, \Delta_{set} \,\rangle$ where $\Delta_{set} = \Delta_{ip^{src}} \cup \Delta_{ip^{dest}} \cup \Delta_{pr} \cup \Delta_{port^{src}} \cup \Delta_{port^{dest}} \cup \Delta_{ste} \cup \{packet(q_{ip^{src}}, q_{ip^{dest}}, q_{pr}, q_{port^{src}}, q_{port^{dest}}, q_{ste}) \to q_F^{set}\}$, q_k being, for any k, the final state of the automaton $\mathbb{A}_k$ where $\mathbb{A}_{ip^{src}} = \mathbb{A}_{[ip_1^{src}, ip_2^{src}]}$, $\mathbb{A}_{ip^{dest}} = \mathbb{A}_{[ip_1^{dest}, ip_2^{dest}]}$, $\mathbb{A}_{pr} = (\Sigma_{firewall}, \{q_{pr}\}, \{q_{pr}\}, \Delta_{pr})$ with $\Delta_{pr} = \{\rho \to q_{pr} \mid \rho \in pr\}$, $\mathbb{A}_{port^{src}} = \sqcap_{1/port_1^{src}}(\mathbb{A}_{port\_inf}) \cap \sqcap_{2/port_2^{src}}(\mathbb{A}_{port\_inf})$, $\mathbb{A}_{port^{dest}} = \sqcap_{1/port_1^{dest}}(\mathbb{A}_{port\_inf}) \cap \sqcap_{2/port_2^{dest}}(\mathbb{A}_{port\_inf})$, $\mathbb{A}_{ste} = (\Sigma_{firewall}, \{q_{ste}\}, \{q_{ste}\}, \Delta_{ste})$ with $\Delta_{ste} = \{state \to q_{ste} \mid state \in st\}$ and where Q_{set} contains all states occuring in Δ_{set}.

3.4 Network address translation

In typical cases, network address translation (NAT) simply map a set of addresses to a constant address, *e.g.* $\langle\, 192.168.*.*, 80 \,\rangle \twoheadrightarrow \langle\, 254.128.215.12, 8080 \,\rangle$. In this paper, in order to cover all reasonable implementations of network address translation, we consider the hypothesis where the mapping can build the translated address from the original one, *e.g.* $\langle\, 192.168.x.y, 80 \,\rangle \twoheadrightarrow \langle\, 192.182.x.y, 8080 \,\rangle$. Thus, we assume that a NAT rule is a pair of terms of the form $\langle\, ip, port \,\rangle \twoheadrightarrow \langle\, ip', port' \,\rangle$ where ip, ip' are IP addresses possibly containing variables such that any variable occuring in ip' occurs in ip at the same position, $port$ and $port'$ are either port numbers or a variables such that $port'$ is a variable only if $port$ is the same variable.

3.5 Specification of firewalls

We can therefore define the specification of firewalls we use for performing formal analysis:

[1] set being the tuple $\langle [ip_1^{src}, ip_2^{src}],\ [ip_1^{dest}, ip_2^{dest}],\ pr,\ [port_1^{src}, port_2^{src}],\ [port_1^{dest}, port_2^{dest}], st \rangle$.

Definition 1 (Formal firewall). *A* formal firewall *(or simply* firewall*) is a* 5*-tuple:* $\mathfrak{F} = \langle\, \mathcal{R}_{pre}, \mathbb{A}_{\text{INPUT}}, \mathbb{A}_{\text{FORWARD}}, \mathbb{A}_{\text{OUTPUT}}, \mathcal{R}_{post} \,\rangle$ *where:*

- $\mathcal{R}_{pre}$ *and* $\mathcal{R}_{post}$ *are linear ordered rewrite systems over* $\Sigma_{firewall}$ *in which any rule* $lhs \to rhs$ *is such that* lhs *and* rhs *are of sort packet and for any position* ω *such as* $rhs(\omega) \in \mathcal{X}$*, we have* $lhs(\omega) = rhs(\omega)$*,*
- $\mathbb{A}_{\text{INPUT}}, \mathbb{A}_{\text{FORWARD}}$ *and* $\mathbb{A}_{\text{OUTPUT}}$ *are tree automata over* $\Sigma_{firewall}$ *recognizing a subset of* $\mathcal{T}^{Packet}_{\Sigma_{firewall}}$*.*

Taking advantage of the specification of packet filters and address translations described in the previous sections, we will explain how to automaticaly translate a usual specification of a firewall into a formal one corresponding to the definition given above. For any chain CHAIN and its associated ordered set $Rules_{\text{CHAIN}} = \{\langle\, set_i, decision_i \,\rangle \mid i \in [1, n_{\text{CHAIN}}]\}$, we build $\mathbb{A}_{\text{CHAIN}}$ as follows:

$$\bigcup_{\substack{i=1 \\ decision_i = \text{ACCEPT}}}^{n_{\text{CHAIN}}} \left(rec(set_i) \setminus \left(\bigcup_{k=1}^{i-1} rec(set_k) \right) \right)$$

Contrary to the three filter chains of the firewall which are translated in the same way, NAT rules are translated differently depending on the target (pre- or postrouting). Indeed, prerouting translates destination address of packets while postrouting translates source ones. Thus, to any NAT rule $\langle\, ip_1, port_1 \,\rangle \twoheadrightarrow \langle\, ip_2, port_2 \,\rangle$ we associate the pair of first order terms:

$$\begin{pmatrix} packet(x_{ip^{src}}, \overrightarrow{ip_1}, x_{pr}, x_{port^{src}}, \overrightarrow{port_1}, x_{ste}), \\ packet(x_{ip^{src}}, \overrightarrow{ip_2}, x_{pr}, x_{port^{src}}, \overrightarrow{port_2}, x_{ste}) \end{pmatrix}$$

in the prerouting case, where $\overrightarrow{ip_i}$ is the first order term representation of ip_i, *i.e.* if $ip_i = octet_1.octet_2.octet_3.octet_4$ then $\overrightarrow{ip_i} = ip(\overrightarrow{octet_1}, \overrightarrow{octet_2}, \overrightarrow{octet_3}, \overrightarrow{octet_4})$ else $\overrightarrow{ip_i} = ip_i \in \mathcal{X}$. The representation of $port_i$ as a term is obtained in the same way. In the postrouting case, we build the pair $packet(\overrightarrow{ip_1}, x_{ip^{dest}} x_{pr}, \overrightarrow{port_1}, x_{port^{dest}}, x_{ste})$ and $packet(\overrightarrow{ip_2}, x_{ip^{dest}} x_{pr}, \overrightarrow{port_2}, x_{port^{dest}}, x_{ste})$. We then obtain for prerouting (resp. postrouting) an ordered list of pairs of first order terms which corresponds to $\mathcal{R}_{pre}$ (resp. $\mathcal{R}_{post}$).

Example 1. An example of firewall is given in Figure 1. Since it would be useless to give the whole corresponding formal firewall, we give only a part of $\mathbb{A}_{\text{INPUT}}$ and $\mathcal{R}_{pre}$: $\mathbb{A}_{\text{INPUT}} = (\Sigma_{firewall}, Q_{\text{INPUT}}, \{q_F\}, \Delta_{\text{INPUT}})$ with

$$\Delta_{\text{INPUT}} = \begin{cases} packet(q_1^1, q_2, q_3, q_4, q_5, q_6) \to q_F & tcp \to q_3 \\ packet(q_1^2, q_2, q_3, q_4, q_5, q_6) \to q_F & \overrightarrow{80} \to q_4 \\ new \to q_6 & \dots \end{cases}$$

where q_1^1 is the final state of $\mathbb{A}_{[192.168.10.0,192.168.10.100]}$, q_1^2 is the final state of $\mathbb{A}_{[192.168.10.102,192.168.10.255]}$, q_2 the one of Ω_{IP} and q_5 the one of Ω_{Port}.
$\mathcal{R}_{pre}$ consists in:

$$\begin{cases} packet(x_1, \overrightarrow{121.130.1.x}, x_2, x_3, \overrightarrow{5222}, x_4) \to packet(x_1, \overrightarrow{121.130.1.x}, x_2, x_3, \overrightarrow{777}, x_4) \\ packet(x_1, \overrightarrow{216.239.37.125}, x_2, x_3, \overrightarrow{5222}, x_4) \to packet(x_1, \overrightarrow{195.20.241.122}, x_2, x_3, \overrightarrow{777}, x_4) \end{cases}$$

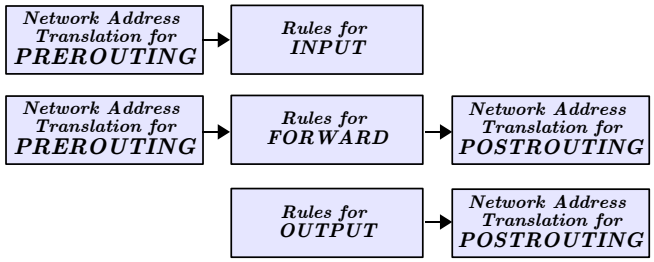
Chain	*IP address source*	*IP address destination*	*Protocol*	*Port source*	*Port destination*	*State*	*Decision*
INPUT	192.168.10.101	$*.*.*.*$	TCP	80	*any*	*any*	DROP
INPUT	192.168.10.$*$	$*.*.*.*$	TCP	80	*any*	*any*	ACCEPT
INPUT	$*.*.*.*$	$*.*.*.*$	*any*	*any*	*any*	*any*	DROP
OUTPUT	$*.*.*.*$	192.168.10.$*$	TCP	80	*any*	*any*	ACCEPT
OUTPUT	$*.*.*.*$	$*.*.*.*$	*any*	*any*	*any*	*any*	*deny*
FORWARD	192.168.20.$*$	121.130.$*.*$	TCP	80	*any*	*any*	ACCEPT
FORWARD	$*.*.*.*$	$*.*.*.*$	TCP	*any*	$\{5222, 443\}$	*any*	DROP
FORWARD	192.168.20.$*$	209.85.229.104	TCP	80	*any*	*any*	DROP
FORWARD	$*.*.*.*$	$*.*.*.*$	TCP	80	*any*	*any*	ACCEPT
FORWARD	$*.*.*.*$	$*.*.*.*$	*any*	*any*	*any*	*any*	DROP

$$\begin{aligned} \textit{Prerouting } NAT: \quad & \langle\, 121.130.1.x, 5222 \,\rangle \twoheadrightarrow \langle\, 121.130.1.x, 777 \,\rangle \\ & \langle\, 216.239.37.125, 5222 \,\rangle \twoheadrightarrow \langle\, 195.20.241.122, 777 \,\rangle \\ \textit{Postrouting } NAT: \quad & \langle\, 192.168.2.38, x \,\rangle \twoheadrightarrow \langle\, 195.20.241.122, x \,\rangle \end{aligned}$$

Fig. 1. Example of firewall

4 Semantics and equivalence

Starting from the formal specification of firewalls given in the previous section, we are now interested in giving a semantics to firewalls in order to compare them. For that, we extract from the general architecture of firewalls given in Section 3 all possible flows of packets:



Indeed, given a packet and a firewall, only three cases are possible:

- the packet is intended to the firewall itself, then it will pass through prerouting and the input chain;
- the destination of the packet is not the firewall, and then it will cross the prerouting, the forward chain and the postrouting;
- or the packet is sent out by the firewall, then it will go across the output chain and the prerouting.

We then study the semantics of the traversal of each of these three possible paths. We write $pkt_1 \rightarrow_{pre} pkt_2$ (resp. $pkt_1 \rightarrow_{post} pkt_2$) iff pkt_2 is obtained by applying the first applicable rule of $\mathcal{R}_{pre}$ (resp. $\mathcal{R}_{post}$) from pkt_1 or $pkt_2 = pkt_1$ if no rule is applicable and we say that a firewall $\mathfrak{F}$ *accepts* a packet pkt :

- in the INPUT chain iff $pkt \rightarrow_{pre} pkt' \in \mathcal{L}(\mathbb{A}_{\text{INPUT}})$;
- in the FORWARD chain iff $pkt \rightarrow_{pre} pkt' \in \mathcal{L}(\mathbb{A}_{\text{FORWARD}})$;
- in the OUTPUT chain iff $pkt \in \mathcal{L}(\mathbb{A}_{\text{OUTPUT}})$;

for some pkt'. If $\mathfrak{F}$ does not accept pkt in the chain CHAIN, we say that $\mathfrak{F}$ drops pkt in CHAIN. Moreover, we say that $\mathfrak{F}$ *delivers a packet* pkt *under the form* pkt' *through* :

- INPUT iff $\mathfrak{F}$ accepts pkt in INPUT and $pkt \rightarrow_{pre} pkt'$;
- FORWARD iff $\mathfrak{F}$ accepts pkt in FORWARD and there exists a packet pkt'' such that $pkt \rightarrow_{pre} pkt'' \rightarrow_{post} pkt'$;
- OUTPUT iff $\mathfrak{F}$ accepts pkt in OUTPUT and $pkt \rightarrow_{post} pkt'$.

We define for any chain CHAIN a binary relation $\overset{\text{CHAIN}}{\multimap}_{\mathfrak{F}}$ such that for any packet pkt: $pkt \overset{\text{CHAIN}}{\multimap}_{\mathfrak{F}} pkt'$ iff $\mathfrak{F}$ deliver pkt through CHAIN under the form pkt'. When there is no pkt' such that $pkt \overset{\text{CHAIN}}{\multimap}_{\mathfrak{F}} pkt'$, *i.e.* when the packet is dropped by CHAIN, we write $pkt \overset{\text{CHAIN}}{\multimap}_{\mathfrak{F}} \times$.

Example 2. We still consider the firewall described in Figure 1. We have for example: $packet(\overrightarrow{192.168.10.2}, \overrightarrow{208.68.163.220}, tcp, 5222, 5222, new) \overset{\text{FWD}}{\multimap}_{\mathfrak{F}} \times$ and

$$packet(\overrightarrow{121.130.1.2}, \overrightarrow{216.239.37.125}, tcp, 5222, 5222, new) \overset{\text{FWD}}{\multimap}_{\mathfrak{F}} packet(\overrightarrow{121.130.1.2}, \overrightarrow{195.20.241.122}, tcp, 5222, 777, new)$$

Definition 2 (Semantics of a firewall). *Given a firewall* $\mathfrak{F} = \langle\, \mathcal{R}_{pre}, \mathbb{A}_{\text{INPUT}}, \mathbb{A}_{\text{FORWARD}}, \mathbb{A}_{\text{OUTPUT}}, \mathcal{R}_{post} \,\rangle$, *we call* semantics of $\mathfrak{F}$ *the 3-tuple*

$$[\![\mathfrak{F}]\!] = \left\langle \overset{\text{IN}}{\multimap}_{\mathfrak{F}}, \overset{\text{FWD}}{\multimap}_{\mathfrak{F}}, \overset{\text{OUT}}{\multimap}_{\mathfrak{F}} \right\rangle$$

Definition 3 (Firewall equivalence). *Two firewalls* $\mathfrak{F}$ *and* $\mathfrak{F}'$ *are said to be* equivalent *iff they have the same semantics.*

Theorem 1. *Equivalence between firewalls is decidable.*

In order to prove this theorem, we give an algorithm computing the semantics of a firewall and will show that the semantics of a firewall always belongs to a class of sets in which the equality is decidable.

Lemma 1. *For any firewall* $\mathfrak{F} = \langle\, \mathcal{R}_{pre}, \mathbb{A}_{\text{INPUT}}, \mathbb{A}_{\text{FORWARD}}, \mathbb{A}_{\text{OUTPUT}}, \mathcal{R}_{post} \,\rangle$, $\rightarrow_{pre}$ *and* $\rightarrow_{post}$ *are recognizable relations.*

Proof. Let be a rule $lhs \rightarrow rhs$ such that $lhs, rhs : Packet$ and $\forall \omega \in \mathcal{P}os(rhs)$, if $rhs(\omega) \in \mathcal{X}$ then $lhs(\omega) = rhs(\omega)$. The set of pairs of ground terms pkt_1, pkt_2 such that $pkt_1 \rightarrow_{\mathcal{R}} pkt_2$ can be denoted by the following tree automaton $\mathbb{A}[lhs \rightarrow rhs] = (\Sigma_{firewall}, Q, \{q_F\}, \Delta)$ where Δ contains:

- $\langle\, packet, packet \,\rangle\, (q_1, q_2, q_3, q_4, q_5, q_6) \rightarrow q_F$;

- for any $\omega \in \mathcal{P}os(lhs)$ *s.t.* $lhs(\omega) \notin \mathcal{X}$, $\langle\, lhs(\omega), rhs[\omega)\,\rangle\,(q_{\omega.1}, \ldots, q_{\omega.n}) \rightarrow q_\omega$ with $n = \max(ar(lhs(\omega)), ar(rhs[\omega))$;
- for any $\omega \in \mathcal{P}os(rhs) \setminus \mathcal{P}os(lhs)$, $\langle\, \Lambda, rhs(\omega)\,\rangle\,(q_{\omega.1}, \ldots, q_{\omega.n}) \rightarrow q_\omega$ with $n = ar(rhs(\omega))$;
- for any $\omega \in \mathcal{P}os(lhs)$ *s.t.* $lhs(\omega)$ is a variable of sort s and $rhs(\omega)$ is not a variable, the set of rules of the automaton $\mathbb{A}^s_{sort} \times rec\left(\{rhs_{|\omega}\}\right)$ whose final state is renamed into q_ω;
- for any ω *s.t.* $rhs(\omega) = lhs(\omega)$ is a variable of sort s, the set of rules of the automaton Id^2_s;

and Q contains all states occuring in Δ.
For any rewrite system $\mathcal{R}$ containing an ordered set of rules of the form described above, we denote by $\mathbb{A}_\mathcal{R}$ the following automaton:

$$\bigcup_{i=1}^{n} \mathbb{A}[lhs_i \rightarrow rhs_i] \setminus \left(\bigcup_{k=1}^{i-1} (rec(lhs_k) \times rec(rhs_i)) \right)$$

The automaton recognizing $\rightarrow_{pre}$ (resp. $\rightarrow_{post}$) is the one denoted by $\mathbb{A}_{pre}$ (resp. $\mathbb{A}_{post}$) such that: $\mathbb{A}_{pre} = \mathbb{A}_{\mathcal{R}_{pre}} \cup \left(Id^2_{Packet} \setminus \left(\sqcap_1(\mathbb{A}_{\mathcal{R}_{pre}}) \times \sqcap_1(\mathbb{A}_{\mathcal{R}_{pre}})\right)\right)$ (resp. $\mathbb{A}_{post} = \mathbb{A}_{\mathcal{R}_{post}} \cup \left(Id^2_{Packet} \setminus \left(\sqcap_1(\mathbb{A}_{\mathcal{R}_{post}}) \times \sqcap_1(\mathbb{A}_{\mathcal{R}_{post}})\right)\right)$). Note that $Id^2_{Packet} \setminus (\sqcap_1(\mathbb{A}_k) \times \sqcap_1(\mathbb{A}_k))$ represents the identity relation whose domain is all packets except the ones belonging to the domain of the relation denoted by $\mathbb{A}_k$.

Proposition 1. *For any firewall* $\mathfrak{F} = \langle\, \mathcal{R}_{pre}, \mathbb{A}_{\text{INPUT}}, \mathbb{A}_{\text{FORWARD}}, \mathbb{A}_{\text{OUTPUT}}, \mathcal{R}_{post}\,\rangle$, *We have:*

$$\begin{aligned}
\overset{\text{IN}}{\rightsquigarrow}_\mathfrak{F} &= \mathcal{L}\left(\mathbb{A}_{pre} \cap (\sqcup_1(\mathbb{A}_{\text{INPUT}}))\right) \\
\overset{\text{FWD}}{\rightsquigarrow}_\mathfrak{F} &= \mathcal{L}\left(\sqcap_2\left(\sqcup_3(\mathbb{A}_{pre}) \cap \sqcup_{1,3}(\mathbb{A}_{\text{FORWARD}}) \cap \sqcup_1(\mathbb{A}_{post})\right)\right) \\
\overset{\text{OUT}}{\rightsquigarrow}_\mathfrak{F} &= \mathcal{L}\left(\sqcup_2(\mathbb{A}_{\text{OUTPUT}}) \cap \mathbb{A}_{post}\right)
\end{aligned}$$

This proposition indicates that the semantics of any firewall is a 3-tuples of recognizable sets. Since the equality of recognizable sets is decidable ([CDG+08,Jac96]), we obtain the decidability of equivalence of firewalls.

5 Analysis of firewalls

5.1 Detecting misconfiguration

A lot of recent works [ASP00], [ASHBH05], [BKLR06], [CCBGA05], [GL04], [Liu08], [UC04], [LG05] focus its activities on detecting conflicts (or misconfigurations) in firewalls. Most of them developped algorithms based on properties over natural numbers intervals or decision diagrams. Both techniques do not allow to deal with address translation nor resolving queries of the kind "Who can access to google.com ?". We will show in this section that our framework allows to solve all decision problems raised in the litterature (shadowing, redundancy, compactness, completeness) as well as more complex problems.

Definition 4 ([ASHBH05,CCBGA05,YMS+06] Shadowing). *We say that a firewall has* shadowing *iff it contains at least one filtering rule such that all packets it accepts (resp. drops) are dropped (resp. accepted) by a prior rule. In such a case, the concerned rule is said to be* shadowed.

For any chain CHAIN, the ith rule $\langle\, set_i, decision_i \,\rangle$ is shadowed iff the following automaton recognizes the empty language:

$$rec(set_i) \setminus \left(\bigcup_{\substack{k=1 \\ decision_k \neq decision_i}}^{i-1} rec(set_k) \right)$$

Definition 5 ([ASHBH05,CCBGA05,YMS+06,LG05] Redundancy). *We say that a firewall has* redundancy *iff it contains at least one filtering rule which is not shadowed by any other rule but which can be removed without changing the filtering result.*

This definition of redundancy is relevant only if we consider only packet filtering. Indeed, address translation can make a rule useless even if there is no redundancy. That is why we prefer to talk about *uselessness* instead of redundancy. This being so, since the equivalence of firewalls is decidable, it is sufficient for checking the uselessness of a filtering rule r to compute the semantics of $\mathfrak{F}_{[r]}$ where $\mathfrak{F}_{[r]}$ is the firewall in which the rule r has been removed and to verify if $[\![\mathfrak{F}_{[r]}]\!] = [\![\mathfrak{F}]\!]$. However, we can see that this is not necessary. Indeed, a rule r is uselessness if either the union of prior rules matches all packets which can be matched by r or when r matches packets that the prerouting transforms into another packet. The first case is equivalent to check for non shadowed rules $\langle\, set_i, decision_i \,\rangle$ if the set $rec(set_i) \setminus \left(\bigcup_{k=1}^{i-1} rec(set_k)\right)$ is empty. For the second case, it is sufficient to compute the set of packets which can be at the output of the prerouting process. Since it is the set of packets pkt' such that there exists another packet pkt such as $\langle\, pkt, pkt' \,\rangle \in \mathbb{A}_{pre}$, the set in question is recognized by $\sqcap_1(\mathbb{A}_{pre})$. Thus, any non shadowed rule $\langle\, set_i, decision_i \,\rangle$ such that $rec(set_i) \setminus \left(\bigcup_{k=1}^{i-1} rec(set_k)\right) \neq \varnothing$ is *bypassed* by the prerouting process iff the automaton $\sqcap_1(\mathbb{A}_{pre}) \cap rec(set_i) \setminus \left(\bigcup_{k=1}^{i-1} rec(set_k)\right)$ recognizes the empty set. [GL04] pointed out another possible misconfiguration called completness. The aim of this property is to check if a firewall always take a decision for any input packet.

Definition 6 ([GL04] Completeness). *A firewall is said* complete *iff every packet satisfies at least one filtering rule of the firewall.*

To check that this property holds for a firewall $\mathfrak{F}$, we compute the set of packets for which the firewall is able to take a decision, that is to say the set of $pkt : Packet$ such as $\exists pkt', \langle\, pkt, pkt' \,\rangle \in \mathcal{L}(\mathbb{A}_{pre})$ and $pkt' \in \mathcal{L}\left(\bigcup_{i=1}^{n_{\text{INPUT}}} rec(set_i)\right)$

or $pkt' \in \mathcal{L}\left(\bigcup_{i=1}^{n_{\text{FORWARD}}} rec(set_i)\right)$, which is a recognizable set denoted by:

$$\sqcap_2 \left(\mathbb{A}_{pre} \cap \sqcup_1 \left(\bigcup_{i=1}^{n_{\text{INPUT}}} rec(set_i) \cup \bigcup_{i=1}^{n_{\text{FORWARD}}} rec(set_i) \right) \right)$$

We have thus to verify that the language denoted by this automaton is $\mathcal{T}^{Packet}_{\Sigma_{firewall}}$ (which is decidable).

5.2 Properties checking

We are now interested in checking other properties looking like queries in databases: *e.g.* "Who can access to google.com" ? First of all, we need to express this query in a formal way: we search the set of IP x_{ip} such that every packet pkt of the form $packet(x_{ip}, x_1, x_2, x_3, 80, x_4)$ is delivered by FORWARD under the form $packet(y, \overrightarrow{209.85.229.104}, tcp, y_1, 80, y_2)$ such that packets of the form $packet(\overrightarrow{209.85.229.104}, y, tcp, 80, y_1, z_2)$ are delivered by FORWARD under the form $packet(w_1, x_{ip}, tcp, w_2, 80, w_3)$. To solve this query, we need the following lemma:

Lemma 2 (Decomposition lemma). *Given a tree automaton $\mathbb{A}$ over a signature Σ, if there exists a $f \in \Sigma$ such that for any $t \in \mathcal{L}(\mathbb{A})$, $t_{|\varepsilon} = f$, then there exists an automaton denoted by $\partial_f(\mathbb{A})$ such that $\langle t_1, \ldots, t_n \rangle \in \mathcal{L}(\partial_f(\mathbb{A}))$ iff $f(t_1, \ldots, t_n) \in \mathcal{L}(\mathbb{A})$.*

By denoting by $\mathbb{A}_{tr} = \partial_{\langle packet, packet \rangle}\left(\overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}}\right)$, the query corresponds to the search of the set of x_{ip} such that $\forall x_1, x_2, x_3, x_4, \exists y, y_1, y_2, \forall z_1, \exists w_1, w_2, w_3$:

$$\left\langle x_{ip}, y, x_1, \overrightarrow{209.85.229.104}, x_2, tcp, x_3, y_1, 80, 80, x_4, y_2 \right\rangle \in \mathcal{L}(\mathbb{A}_{tr}) \text{ and}$$
$$\left\langle \overrightarrow{209.85.229.104}, w_1, y, x_{ip}, tcp, tcp, 80, w_2, y_1, 80, z_1, w_3 \right\rangle \in \mathcal{L}(\mathbb{A}_{tr})$$

Details of the resolution of this query is not given but follows the algorithm described in [BC10]. The solution is $\mathcal{L}(\alpha)$ where:

- $\alpha_0 = \sqcup_J(\mathbb{A}_{tr}) \cap \sqcup_I(\mathbb{A}_{tr}) \cap \sqcup_{K_1}(Id^2_{IP}) \cap \sqcup_{K_2}(Id^2_{IP}) \cap \sqcup_{K_3}(Id^2_{port})$ where $I = [1, 12]$, $J = [13, 24]$, $K_1 = [1, 24] \setminus \{1, 16\}$, $K_2 = [1, 24] \setminus \{2, 15\}$ and $K_3 = [1, 24] \setminus \{8, 21\}$;
- $\alpha_1 = \sqcap_{4/\overrightarrow{209.85.229.104},6/tcp,9/80,10/80,1/\overrightarrow{209.85.229.104},5/tcp,6/tcp,7/80,10/80}(\alpha_0)$;
- $\alpha_2 = \sqcap_{14,20,24,16,15,21}(\alpha_1)$; $\alpha_3 = \overline{\sqcap_{23}(\overline{\alpha_2})}$; $\alpha_4 = \sqcap_{2,8,12}(\alpha_3)$ and
- $\alpha = \overline{\sqcap_{3,5,7,4}(\overline{\alpha_4})}$.

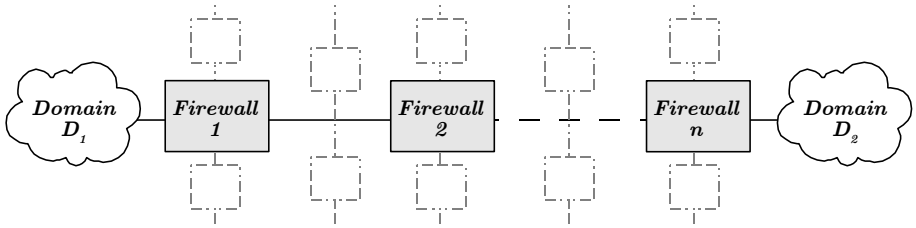
Beyond this example, what is interesting to remark is that our framework allows us to check properties which cannot be checked by covering every path and every packet since there would be in worst case about $2^{154} > 2.10^{46}$ possibilities[2]

[2] At the rate of 10^9 instructions per second, considering that a case is explored in one instruction, it would take more than 6.10^{26} millenniums.

to explore. In the same way, we could compute the set of addresses a computer can reach or others more complex queries, which is very usefull when addressing the problem of verifying the behaviour of firewalls whithout exhaustively testing them.

5.3 Networks of firewalls

In this section we will show that our framwork allows us to extend analysis methods previously presented to networks of firewalls. Thus, we focus our attention on traversal flows, in other terms, we study the part of firewalls consisting in the sequence of prerouting, FORWARD chain filtering and postrouting and analyse compositions of such sequences. The aim of this paper being mainly to show the perspectives opened by the specification of firewalls using tree automata, we will not study in depth the composition of firewalls but only show over a simple case of composition that our framework is suitable. Thus, let us consider the problem of analysing traffic between two domains separated by n firewalls in series:



As the figure lets us guess, we do not care about the flows of packets which do not come from D_1 and whose destination is not D_2. To do this, we should modelize the routing functionality of firewalls, which is not the purpose of this paper (but which is interesting and feasible). Then, a good advantage of using tree automata to modelize the semantics of firewalls is that the semantics of a sequence of firewalls is very natural:

Definition 7 (Semantics of a sequence of firewalls). *Given n firewalls $\mathfrak{F}_1, \ldots, \mathfrak{F}_n$, the semantics of the sequence $\langle \mathfrak{F}_1, \ldots, \mathfrak{F}_n \rangle$ is the pair of relations:*

$$[\![\mathfrak{F}_1 \oplus \ldots \oplus \mathfrak{F}_n]\!] = \left\langle \overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_1,\ldots,\mathfrak{F}_n}, \overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_n,\ldots,\mathfrak{F}_1} \right\rangle$$

such that:

$$\begin{cases} \overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_1,\ldots,\mathfrak{F}_n} = \overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_1,\ldots,\mathfrak{F}_{n-1}} \circ \overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_n} \\ \overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_1} \circ \overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_2} = \mathcal{L}\left(\sqcap_2\left(\sqcup_3\left(\mathbb{A}^{\text{FORWARD}}_{\mathfrak{F}_1}\right) \cap \sqcup_1\left(\mathbb{A}^{\text{FORWARD}}_{\mathfrak{F}_2}\right)\right)\right) \end{cases}$$

where $\mathbb{A}^{\text{FORWARD}}_{\mathfrak{F}_i} = \sqcap_2\left(\sqcup_3(\mathbb{A}_{pre}) \cap \sqcup_{1,3}(\mathbb{A}_{\text{FORWARD}}) \cap \sqcup_1(\mathbb{A}_{post})\right)$ *with* $\mathfrak{F}_i = \langle \mathcal{R}_{pre}, \mathbb{A}_{\text{INPUT}}, \mathbb{A}_{\text{FORWARD}}, \mathbb{A}_{\text{OUTPUT}}, \mathcal{R}_{post} \rangle$.

The relation $\overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_1,\ldots,\mathfrak{F}_n}$ (resp. $\overset{\text{FWD}}{\rightsquigarrow}_{\mathfrak{F}_n,\ldots,\mathfrak{F}_1}$) corresponds to a function which associates to any packet whose origin belongs to D_1 (resp. D_2) and whose destination belongs to D_2 (resp. D_1) the form of the packet under which it will be delivered after the traversal of the firewalls sequence. A packet has no image if it is dropped during its traversal or redirected toward another domain. Thus, it is very easy to check (since equivalence between recognizable tree languages is decidable) if severals sequences of firewalls have the same behaviour, *i.e.* generate the same set of allowed traffics between D_1 and D_2. Moreover, the properties checking is naturally extended to a sequence of firewalls.

6 Conclusion and future work

We have proposed in this paper an original way to specify firewalls using tree automata and we have shown that this specification allows us to extend analysis usually performed only on packet filtering to a more real representation of firewalls. We have also proved that our framework allows us to deal with more complex questions over firewalls. We currently work on three follow-ups of the results presented in this paper. The first one concerns composition of firewalls: we must modelize the routing fonctionality of firewalls in order to define the semantics of their composition within a network. Indeed, it would be interesting to analyse reachability in complex networks of firewalls or to detect firewalls which can be removed whithout changing the traffic. The second possible progress is the development of an algorithm which builds a firewall from a semantics of firewall, which would allow us to address the problem of minimalizing firewalls, *i.e.* build from a firewall another equivalent one with a minimum of filtering and NAT rules, and the problem of merging firewalls when it is possible. Finally, we are currently attached to develop a tool in order to interrogate firewalls as databases. Indeed, the properties checking is very closed to deductive databases queries and it would be usefull to have a query language allowing us to request user defined statements on a firewall network.

Acknowledgment

This work was supported by ANR SSURF and région Lorraine.

References

ASHBH05. E Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan. Conflict classification and analysis of distributed firewall policies. In *IEEE Journal on Selected Areas in Communications*, volume 23, pages 2069 – 2084, 2005.

ASP00. H. Adiseshu, S. Suri, and G. Parulkar. Detecting and resolving packet filter conflicts. In *19th Annual Joint Conference of the IEEE Computer and Communication Societies*, pages 1203–1212, 2000.

BC10. T. Bourdier and H. Cirstea. Constrained rewriting in recognizable theories, 2010. Submitted.

BCJK09. T. Bourdier, H. Cirstea, M. Jaume, and H. Kirchner. On Formal Specification and Analysis of Security Policies. *http://hal.inria.fr/inria-00429240/en/*, 2009. Submitted to the Journal of automated reasoning.

BKLR06. A.K. Bandara, A. Kakas, E.C. Lupu, and A. Russo. Using argumentation logic for firewall policy specification and analysis. In *17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management Large Scale Management (DSOM)*, pages 185–196, 2006.

BN98. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

CCBGA05. F. Cuppens, N. Cuppens-Boulahia, and J. Garcia-Alfaro. Detection and removal of firewall misconfiguration. In *International Conference on Communication, Network and Information Security (IASTED)*, 2005.

CDG+08. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2008. release November, 18th 2008.

CF02. B. Conoboy and E. Fictner. Ip filter based firewalls howto. Available on: `http://www.obfuscation.org/ipf/ipf-howto.pdf`, 2002.

GL04. M.G. Gouda and A.X. Liu. Firewall design: Consistency, completeness, and compactness. In *24th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2004.

HASM05. H. Hamed, E Al-Shaer, and W. Marrero. Modeling and verification of ipsec and vpn security policies. In *13th IEEE International Conference on Network Protocols*, 2005.

Jac96. F. Jacquemard. *Automates d'arbres et rcriture de termes*. PhD thesis, Université de Paris-Sud, UFR scientifique d'Orsay, 1996.

KFS+03. S Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen. Analysis of vulnerabilities in internet firewalls. In *Computers & Security*, volume 22, pages 214–232, 2003.

KK06. C. Kirchner and H. Kirchner. Rewriting solving proving. Preliminary version of a book, 2006. `http://www.loria.fr/~hkirchne/rsp.pdf`.

LG05. A.X. Liu and M.G. Gouda. Complete redundancy detection in firewalls. In *Data and Applications Security*, pages 193–206, 2005.

Liu08. A.X. Liu. Formal verification of firewall policies. In *IEEE International Conference on Communications (ICC)*, pages 1494 – 1498, 2008.

Rus02. R. Russell. Linux 2.4 packet filtering howto. Available on: `http://www.netfilter.org/documentation`, 2002.

UC04. T.E. Uribe and S. Cheung. Automatic analysis of firewall and network intrusion detection system configurations. In *Formal Methods In Security Engineering (FMSE)*, 2004.

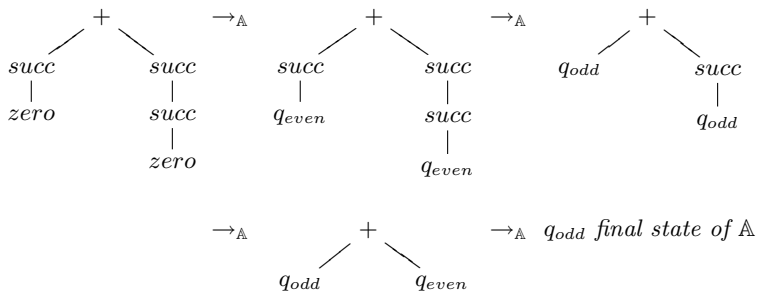
YMS+06. L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra. FIREMAN: A toolkit for firewall modeling and analysis. In *IEEE Symposium on Security and Privacy (S&P)*, 2006.

A Examples of tree automata

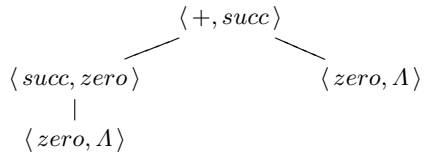
Let be Σ the signature containing $+ : Nat \mapsto Nat$, $zero : \mapsto Nat$ and $s : Nat \mapsto Nat$. The following unary automaton $\mathbb{A} = (\Sigma, \{q_{odd}, q_{even}\}, \{q_{odd}\}, \Delta)$ where Δ consists of: $zero \to q_{even}$ and

$+(q_{odd}, q_{odd}) \to q_{even}$ $\quad +(q_{even}, q_{even}) \to q_{even}$ $\quad succ(q_{even}) \to q_{odd}$
$+(q_{odd}, q_{even}) \to q_{odd}$ $\quad +(q_{even}, q_{odd}) \to q_{odd}$ $\quad succ(q_{odd}) \to q_{even}$

recognizes the set of odd numbers in peano representation. The term $succ(zero)+succ(succ(zero))$ is in $\mathcal{L}(\mathbb{A})$:



The behaviour of n-ary automata is the same except that nodes of configurations are n-tuples. For example, to know if the pair $\langle\, succ(zero) + zero, succ(zero) \,\rangle$ is recognized by an automaton, we must build the following initial configuration:



B Arithmetic on binary numbers

$\mathbb{A}_{bin\_add} = \{(\Sigma_{firewall} \cup \{\Lambda\})^3, Q, \{q_0\}, \Delta_{Add}\}$ where $Q = \{q_0, q_1, q_{000}, \ldots, q_{111}\}$ and

$$\Delta_{Add} = \begin{cases}
\langle 0,0,0 \rangle \to q_{000} & \langle 0,0,1 \rangle \to q_{001} \\
\langle 0,1,0 \rangle \to q_{010} & \langle 0,1,1 \rangle \to q_{011} \\
\langle 1,0,0 \rangle \to q_{100} & \langle 1,0,1 \rangle \to q_{101} \\
\langle 1,1,0 \rangle \to q_{110} & \langle 1,1,1 \rangle \to q_{111} \\
\langle bit,bit,bit \rangle (q_{000}) \to q_0 & \langle bit,bit,bit \rangle (q_{101}) \to q_0 \\
\langle bit,bit,bit \rangle (q_{011}) \to q_0 & \langle bit,bit,bit \rangle (q_{110}) \to q_1 \\
\langle bin,bin,bin \rangle (q_{000},q_0) \to q_0 & \langle bin,bin,bin \rangle (q_{001},q_1) \to q_0 \\
\langle bin,bin,bin \rangle (q_{011},q_0) \to q_0 & \langle bin,bin,bin \rangle (q_{010},q_1) \to q_1 \\
\langle bin,bin,bin \rangle (q_{101},q_0) \to q_0 & \langle bin,bin,bin \rangle (q_{100},q_1) \to q_1 \\
\langle bin,bin,bin \rangle (q_{110},q_0) \to q_1 & \langle bin,bin,bin \rangle (q_{111},q_1) \to q_1
\end{cases}$$