Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 234 (2009) 55–75
www.elsevier.com/locate/entcs

Analysis of Rewrite-Based Access Control Policies

Claude Kirchner, Hélène Kirchner

INRIA
Bordeaux - Sud-Ouest Research Center
351, Cours de la libération, 33405 Talence, France

Anderson Santana de Oliveira[1]

Universidade Federal do Rio Grande do Norte
DIMAp - Campus Universitário - Lagoa Nova
59072-970, Natal-RN, Brazil

Abstract

The rewrite-based approach provides executable specifications for security policies, which can be independently designed, verified, and then anchored on programs using a modular discipline. In this paper, we describe how to perform queries over these rule-based policies in order to increase the trust of the policy author on the correct behavior of the policy. The analysis we provide is founded on the strategic narrowing process, which provides both the necessary abstraction for simulating executions of the policy over access requests and the mechanism for solving *what-if* queries from the security administrator. We illustrate this general approach by the analysis of a firewall system policy.

Keywords: Security Policies, Term Rewriting Systems, strategic rewriting, strategic narrowing.

1 Introduction

Security policies define what it means to be secure for a system. Policies establish constraints on how data are to be accessed, either by determining acceptable information flows, or by describing the privileges principals have over the protected resources inside the system. Since policies reflect the security requirements for some organization, or generally speaking, for some entity, they are subject to frequent changes as new threats appear or as the architecture of the system in question evolves. One of the current challenges in computer security is to model rich, expressive policies, usually given as a set of rules, such that their properties can be

[1] Supported by CNPq 152373/2007-1.

doi:10.1016/j.entcs.2009.02.072

formally stated and proved. For instance, policies should be non-ambiguous, which means that an access is not granted and denied at the same time. Policies should also cover all relevant situations a system may be exposed to: when dealing with access control, this means answering all possible access requests.

In [14,10] we proposed a formal model of policies, based on term rewriting, which provides several advantages: first, the language allows us to handle a wide range of security policies, because we can easily describe the form of the access requests and the set of possible authorization decisions, without restricting them to simply *permit* or *deny*. Moreover, policy application can be defined in a precise and expressive way, since it is possible to determine a strategy to control rule application.

Such an approach provides not only a clear semantics to access control policies, but also appropriate techniques to verify important properties, relying on the confluence, termination and sufficient completeness of the underlying rewrite systems. Moreover, the rewrite-based framework inherits from the modularity results for all these properties, which is a striking advantage over other frameworks, since one can reason about the impact of the policy composition over the properties of the component policies.

Besides proving these properties, a policy designer needs to understand how access decisions are generated. He may want to know how a policy deals with specific kinds of requests, specially those concerning the most sensitive information in the system. In the literature, this is often referred to as "administrator queries" [27,7], representing questions of the kind "what if a request is made under these conditions? Will access be conceded?" Answering these questions increases the confidence of the policy designer in a given specification.

In this paper, we build on our previous works to provide this kind of analysis for rewrite-based policies. The main mechanism behind the analysis is narrowing, which provides both the necessary abstraction for simulation of rewriting-based executions of the policy requests and the solving mechanism of queries. We extend the policy language with requests involving variables and we show how narrowing can be used to find which values can instantiate such variables in order to satisfy given requests, thus providing an adequate mechanism for solving these queries.

Due to the expressiveness and computational power of rewriting, the framework of rewrite-based security policies is general enough to be applied to a large variety of security problems. An illustrative example is provided by firewall policies. A firewall is a security component put in the entry point of a local network to control its interactions with the Internet. The main objective is to inspect the packet traffic from and into the local network and to decide whether a given packet should be transmitted or rejected. Real firewall implementations are usually given as a sequence (i.e. totally ordered set) of rules, which can make recursive calls, and may have default rules. This is the case for instance for NetFilter, which is the firewall system used in several variations of Linux distributions. It is important to notice that such a rule presentation is used as a description of the rule execution sequence. This is indeed an implicit rule execution strategy implemented by these systems, and the capacity of expressing such strategies explicitly is primordial to make the

policy semantics clear, maintainable and provable. In current systems, security administrators often face the problem of avoiding an inconsistent, redundant or incomplete set of rules as described for example in [24].

Here, we show how solving queries can be useful to identify such situations for this extremely important kind of policy, since they are the most current mechanism for protecting networks from numerous threats. But the technique is also applicable for any other form of flexible rewrite-based policy.

The structure of the paper is as follows. In Section 2, we recall and illustrate the definition of a rewrite-based security policy. In Section 3, the relevant definitions of rewriting, narrowing and strategies are given. It stresses the notion of strategic rewriting which is of prime interest in the context of policy rules where the application order and more generally the control are taken into account. Then Section 4 shows how to use strategic narrowing to solve queries and find the corresponding requests in a large class of policy specifications. It provides examples of what-if analysis. To support the reader intuition, a simplified firewall example is developed along the paper. Section 5 concludes with some perspectives for further work.

2 Rewrite-Based Policies

Basic definitions on term rewriting can be found in [29,31,4]. The following standard notations will be used in the following. $\mathcal{T}(\mathcal{F},\mathcal{X})$ is the set of first-order terms built from a given finite set $\mathcal{F}$ of function symbols with their profile declarations and a denumerable set $\mathcal{X}$ of variables. Positions in a term are represented as sequences of integers and denoted by ω, while the empty sequence ϵ denotes the top position. They are ordered by lexicographic prefix ordering $\leq$. The notation $t_{|\omega}$ is used to denote the subterm of t at position ω. We write $t[s]_\omega$ (or sometimes simply $t[s]$) to emphasize that t contains the subterm s at position ω. The set of variables occurring in a term t is denoted by $\mathcal{V}ar(t)$. If $\mathcal{V}ar(t)$ is empty, t is called a *ground term* and $\mathcal{T}(\mathcal{F})$ is the set of ground terms.

A substitution σ is an assignment from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F},\mathcal{X})$, with a finite domain $\{x_1,\ldots,x_k\}$ and written $\sigma = \{x_1 \mapsto t_1,\ldots,x_k \mapsto t_k\}$. The set of variables $\{x_1,\ldots,x_n\}$ is called its domain and denoted $D(\sigma)$. The set $\cup_{i=1,\ldots,n}\mathcal{V}ar(t_i)$ is denoted $I(\sigma)$. We write $\sigma_{|V}$ the restriction of σ to the set of variables V, and $\sigma = \sigma'[V]$ means that $\forall x \in V, \sigma(x) = \sigma'(x)$.

A rewrite rule is an ordered pair of terms $l, r \in \mathcal{T}(\mathcal{F},\mathcal{X})$, denoted $l \to r$, where it is often required that l is not a variable and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The terms l and r are respectively called the left-hand side and the right-hand side of the rule. A rewrite system is a (finite or infinite) set of rewrite rules. Rules can be labeled to easily talk about them.

In [14], we introduced a general definition of an access control policy that abstracts the set of possible requests and decisions, and thus supports a wide range of policies:

Definition 2.1 [Security Policy [14]] An access control security policy $\wp$ is a 5-tuple $(\mathcal{F}, D, R, Q, \zeta)$ where:

(i) $\mathcal{F}$ is a sorted signature; *to define the vocabulary of the data-structure.*

(ii) D is a non-empty set of closed terms: $D \subseteq \mathcal{T}(\mathcal{F})$; *to formalize the set of decisions taken by the policy.*

(iii) R is a set of rewrite rules over $\mathcal{T}(\mathcal{F}, \mathcal{X})$; *to provide the local semantics of the policy.*

(iv) Q is a set of terms from $\mathcal{T}(\mathcal{F})$: $Q \subseteq \mathcal{T}(\mathcal{F})$; *to describe the form of the acceptable requests.*

(v) ζ is a rewrite strategy for R; *to guide the rule application and therefore provide the global semantics of the policy.*

The notions of *local* and *global* semantics mentioned in the comments of this definition are provided as an intuitive way to describe the fact that a rewrite rule is an entity that describes a local transformation and a strategy a way to combine these local application rules. This is formally developed in the next section (section 3).

In order to illustrate this definition, let us consider the following simple example.

Example 2.2 This example is taken from the NetFilter `how-to`[2]. Suppose an Internet user wants to set his firewall to block any traffic coming from the exterior to his local network. Since the interface associated to Internet connections is usually `ppp0`, a simple method is to reject all new packets coming from this source. In order to demonstrate the fact that it might be convenient for a policy to contain rules beyond those which *directly* compute decisions, we also give some additional rules which allow two different local computers to share the same external IP address: for each outgoing packet whose origin is a local machine, its head is rewritten to a single address.

- The sorted signature $\mathcal{F}$ for this policy is:

$pckt$	:	$Address \times Address \times State$	$\mapsto$	$Decision$
$new, estab$	:		$\mapsto$	$State$
$drop, accept$	:		$\mapsto$	$Decision$
$eth0, ppp0, 10.1.1.1,$				
$10.1.1.2, 123.123.1.1$	:		$\mapsto$	$Address$

  The function $pckt$ computes a decision for the packets being transmitted in the network.

- The set of of constant symbols representing decisions is $D = \{accept, drop\}$.
- Consider R as the following rules, where $src, dst : Address$, and $s : State$ are

[2] http://www.netfilter.org

variables:

$$\begin{aligned} pckt(src, dst, estab) &\to accept \\ pckt(eth0, dst, new) &\to accept \\ pckt(ppp0, dst, new) &\to drop \\ pckt(10.1.1.1, ppp0, s) &\to pckt(123.123.1.1, ppp0, s) \\ pckt(10.1.1.2, ppp0, s) &\to pckt(123.123.1.1, ppp0, s) \end{aligned}$$

The first rule says that packets concerning established connections have to be accepted. The second rule matches any packet whose origin is the local network, which are accepted. The third rule rejects any new packet coming from the external network ($ppp0$). The last two rules match the IP address of the local machines and rewrites them to a single IP address visible from the external network.

- The set Q consists in all ground terms having the symbol *pckt* at top position.
- A common strategy for such firewall policy, is to apply rules in the order they are given. We will see how to formalize this strategy in the following section.

The above elements define a security policy. It is worth noticing the presence of recursive rules, which are important for the policy definition, but that do not directly derive permissions.

Further examples to illustrate this definition are given for instance in [14,10,12]. In this framework, policy evaluation corresponds to evaluation by strategic rewriting of ground requests. Let us introduce this concept in the next section.

3 Strategic rewriting and narrowing

Strategic rewriting and narrowing are defined and studied in [28]; the interested reader can refer to it for more details.

Given a rewrite system R, a term t in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ rewrites to another term t' if there exists a rewrite rule $l \to r$ of R, a position ω in t, and a substitution σ such that $t|_\omega = \sigma l$ and $t' = t[\sigma r]_\omega$. This is written $t \to^R_{\omega, l\to r, \sigma} t'$ where either ω, $l \to r$, σ or R may be omitted. $t|_\omega$ is called a redex. A term that has no redex is said to be irreducible for R or to be in R-normal form, or simply normalized. The reflexive transitive closure of the rewriting relation induced by R is denoted by $\overset{*}{\to}^R$.

A term rewrite system R generates an abstract reduction system, i.e. a labeled oriented graph $\mathcal{R} = (\mathcal{O}_R, \mathcal{S}_R)$ whose nodes are terms $\mathcal{O}_R \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, and whose oriented edges are rewriting steps: $\mathcal{S}_R = \{t \to t' | t \to^R_\omega t' \text{ for } \omega \text{ a position in } t\}$. An *$\mathcal{R}$-derivation* or *$\mathcal{R}$-reduction sequence* is a path π in the graph $\mathcal{R}$; when it is finite, π can be written $t_0 \to^{\phi_0} t_1 \to^{\phi_1} t_2 \ldots \to^{\phi_{n-1}} t_n$ where the ϕ_i are labels, and we say that t_0 reduces to t_n by the derivation $\pi = \phi_0\phi_1 \ldots \phi_{n-1}$; this is also denoted $t_0 \to^{\phi_0\phi_1\ldots\phi_{n-1}} t_n$ or simply $t_0 \to^\pi t_n$; n is the *length* of π. A derivation is *empty* when its length is zero, in which case its source and target are the same. The empty derivation issued from t is denoted id_t. The *source* of π is the singleton $\{t_0\}$

denoted $dom(\pi)$. The *target* of π is the singleton $\{t_n\}$ and it is denoted $(\pi\ t_0)$ or simply πt_0 when there is no syntactic ambiguity; note that an $\mathcal{R}$-derivation is the concatenation of its reduction steps. The concatenation of π_1 and π_2 when it exists, is a new $\mathcal{R}$-derivation.

It is common to identify a rewrite system (i.e., a set of rewrite rules) with the abstract reduction system it generates (i.e., the set of all derivations allowed by R). To a rewrite system corresponds directly a unique abstract reduction system that can be seen as a generic way to describe the set of all derivations. The converse is not true since from a set of derivations, the generating rewrite system is not in general uniquely determined.

3.1 Strategic Rewriting

Definition 3.1 [Abstract Strategy] For a given abstract reduction system $\mathcal{R}$:
An *abstract strategy* ζ is a subset of the set of all derivations (finite or not) of $\mathcal{R}$. Applying the strategy ζ on an object t is denoted ζt. It denotes the set of all objects that can be reached from t using a derivation in ζ:

$$\zeta t = \{t' \mid \exists \pi \in \zeta \ such\ that\ t \rightarrow^{\pi} t'\} = \{\pi t \mid \pi \in \zeta\}.$$

When no derivation in ζ has source t, we say that the strategy application on t fails. Applying the strategy ζ on a set of objects consists in applying ζ to each element t of the set. The result is the union of ζt for all t in the set of objects. The *domain* of a strategy is the set of objects that are source of a derivation in ζ: $dom(\zeta) = \bigcup_{\delta \in \zeta} dom(\delta)$. The strategy that contains all the empty derivations is $Id = \{id_t \mid t \in \mathcal{O}\}$.

It is now possible to give the definition of strategic rewriting:

Definition 3.2 [Strategic rewriting [28]] Let $\mathcal{R} = (\mathcal{O}_R, \mathcal{S}_R)$ be the abstract reduction system generated by a rewrite system R and ζ be a strategy of $\mathcal{R}$. A strategic rewriting derivation (or rewriting derivation under strategy ζ) is an element of ζ. A strategic rewriting step under ζ is a rewriting step $t \rightarrow^{R} t'$ that occurs in a derivation of ζ, which is denoted $t \rightarrow^{\zeta} t'$.

A strategy can be described by enumerating all its elements or more suitably by a *strategy language*. Various approaches have been followed, yielding slightly different strategy languages such as ELAN [30,8], Stratego [36], Tom [6,5] [3] or more recently Maude [9]. All these languages share the concern to provide abstract ways to express control of rule applications, by using reflexivity and the meta-level for Maude, or the notion of rewriting strategies for ELAN or Stratego. Strategies such as bottom-up, top-down or leftmost-innermost are higher-order features that describe how rewrite rules should be applied. Tom, ELAN and Stratego provide flexible and expressive strategy languages where high-level strategies are defined by combining low level primitives. We refer to [28] for more details on these strategy languages.

[3] http://tom.loria.fr

In the context of security policies that we want to consider here, a few strategies are of main interest: the strategy $\texttt{universal}(R)$ represents all derivations generated by a set of rewrite rules R, $\texttt{innermost}(R)$ represents derivations where rules of R are applied only on terms whose all strict subterms are irreducible. $\texttt{ordered}(r_1, ..., r_k)$ represents derivations where rules $r_1, ..., r_k$ are tried in this order on terms whose all strict subterms are irreducible. This last strategy corresponds to Innermost Priority rewriting (IP-rewriting for short) that is defined below.

When a rewrite system R is partially ordered with an ordering $\succ$, we say that r_1 has a higher priority than r_2 if r_1 is greater than r_2, denoted $r_1 \succ r_2$. Innermost Priority Rewriting can then be defined as follows:

Definition 3.3 [Innermost Priority Rewriting [21]] Given a rewrite system R with a partial ordering on rules, a term t in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ IP-rewrites to t', if there exists a rewrite rule $l \to r$ of R, a position ω in t, and a substitution σ such that $t|_\omega = \sigma l$ and $t' = t[\sigma r]_\omega$ such that $t \to^R_{\omega, l\to r, \sigma} t'$, no proper subterm of $t|_\omega$ is IP-reducible, and $t|_\omega$ is not reducible by any other rule of higher priority than $l \to r$. This is written $t \to^{IP}_{\omega, l\to r, \sigma} t'$ where either ω, $l \to r$, σ or R may be omitted. $t|_\omega$ is called a IP-redex. A term that has no IP-redex is said to be IP-normalized.

With this definition, we get:

Proposition 3.4 *A term t is IP-normalized iff it is normalized.*

Proof If t is normalized, t is indeed IP-normalized. Let us prove the converse by contraposition. If t is not in normal form, there exist an innermost position and a non empty set of rules that apply at this position. By choosing a rule with the higher-priority between them, we get an IP-rewriting step and t is not IP-normalized. □

A substitution is said (ground) IP-normalized if all the terms in its image are (ground) IP-normalized. According to Proposition 3.4, we may simply consider normalized substitutions.

3.2 The narrowing process

The narrowing process, introduced in [26,16], is quite similar to rewriting but there, matching is replaced by unification. Let us recall its usual definition:

Definition 3.5 [Narrowing] Given a rewrite system R, a term t in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ narrows to the term t' if there exists a rewrite rule $l \to r$ of R and a position ω in t and such that $t|_\omega$ and l are unifiable with a most general unifier (mgu for short) σ. Then $t' = \sigma(t[r]_\omega)$. This is denoted $t \leadsto^R_{\omega, l\to r, \sigma} t'$ or $t \leadsto t'$ when we do not need to make precise which rewrite rule is used and where.

Let us remember that the variables in a rewrite rule are implicitly universally quantified and therefore, the set of variables in a rewrite rule may always be assumed distinct from those in the narrowed term (i.e. in the definition above, $\mathcal{V}ar(t) \cap$

$\mathcal{V}ar(l) = \emptyset$). Let us remark also that narrowing subsumes rewriting since a match between variable disjoint terms is also a unifier.

Based now on the narrowing process, a term rewrite system R generates another abstract reduction system $\mathcal{N} = (\mathcal{O}_R, \mathcal{S}_R)$ with $\mathcal{O}_R \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $\mathcal{S}_R = \{t \rightsquigarrow t' | t \rightsquigarrow^R_{\omega, l \to r} t'$ for ω a position in $t\}$. As for the rewriting relation, we can define strategic narrowing.

Definition 3.6 [Strategic narrowing [28]] Given an abstract reduction system $\mathcal{N} = (\mathcal{O}_R, \mathcal{S}_R)$ generated by a rewrite system R, and a strategy ζ of $\mathcal{N}$, a strategic narrowing derivation (or narrowing derivation under strategy ζ) is an element of ζ. A strategic narrowing step under ζ is a narrowing step $t \rightsquigarrow^R t'$ that occurs in a derivation of ζ, which is denoted $t \rightsquigarrow^\zeta t'$.

3.3 Simulation of rewriting by narrowing

We are now interested in formalizing precisely how (strategic) rewriting and (strategic) narrowing are related. It may be expressed thanks to the general notion of simulation on abstract reduction systems given in [22] and [15] independently.

Definition 3.7 [Simulation] Let $(\mathcal{O}_1, \to_1)$ and $(\mathcal{O}_2, \to_2)$ be two abstract reduction systems. $(\mathcal{O}_1, \to_1)$ Γ-simulates $(\mathcal{O}_2, \to_2)$ for a relation $\Gamma \subseteq \mathcal{O}_1 \times \mathcal{O}_2$ when for every reduction step $s_2 \to_2 s'_2$ with source s_2 and target s'_2, there exists a corresponding reduction step $t_1 \to_1 t'_1$ with source t_1 and target t'_1, such that $t_1 \Gamma s_2$ and $t'_1 \Gamma s'_2$.

Let us define the relation $\Gamma \subseteq \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F}, \mathcal{Y})$ as : $t \Gamma u$ if t is a ground instance of u. The following lemma entails the fact that the abstract reduction system $(\mathcal{T}(\mathcal{F}), \to)$ simulates the abstract reduction system $(\mathcal{T}(\mathcal{F}, \mathcal{Y}), \rightsquigarrow)$.

Lemma 3.8 *For any term $t \in \mathcal{T}(\mathcal{F}, \mathcal{Y})$ and rewrite system R, if $t \rightsquigarrow_{\omega, l \to r, \sigma} t'$ then $\sigma(t) \to_{\omega, l \to r, \sigma} t'$, and for any ground instance α of σ, $\alpha(t) \to_{\omega, l \to r, \sigma} \alpha(t')$.*

Proof Clearly, if $t \rightsquigarrow_{\omega, l \to r, \sigma} t'$ then $\sigma(t) \to_{\omega, l \to r, \sigma} t'$. Since α is an instance of σ, $\alpha = \mu\sigma$. Then $\alpha(t) = \mu(\sigma(t)) \to_{\omega, l \to r} \mu(t')$. But $t' = \sigma(t[r]_\omega)$, so $\mu(t') = \mu\sigma(t[r]_\omega) = \alpha(t')$. □

As a consequence, a derivation in the abstract reduction system $(\mathcal{T}(\mathcal{F}, \mathcal{Y}), \rightsquigarrow)$ corresponds to a set of derivations in the abstract reduction system $(\mathcal{T}(\mathcal{F}), \to)$:

Proposition 3.9 *For any term $t \in \mathcal{T}(\mathcal{F}, \mathcal{Y})$ and rewrite system R, if $t \rightsquigarrow^R_{\omega_1, l_1 \to r_1, \sigma_1} t_1 \ldots \rightsquigarrow^R_{\omega_n, l_n \to r_n, \sigma_n} t_n$, then*

$$\sigma_n \ldots \sigma_1(t) \to^R_{\omega_1, l_1 \to r_1} \ldots \to^R_{\omega_n, l_n \to r_n} t_n$$

and for any ground instance α of $\sigma_n \ldots \sigma_1(t)$,

$$\alpha(t) \to^R_{\omega_1, l_1 \to r_1} \ldots \to^R_{\omega_n, l_n \to r_n} \alpha(t_n).$$

Given a terminating rewrite system R, let us consider $\mathcal{N}$, the subset of normalized ground instances of terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. The relation $\Gamma_\mathcal{N} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{Y}) \times \mathcal{N}$ is

defined as : $u\,\Gamma_{\mathcal{N}}\,t$ if t is a normalized ground instance of u. The fact that the abstract reduction system $(\mathcal{T}(\mathcal{F},\mathcal{Y}),\rightsquigarrow)$ $\Gamma_{\mathcal{N}}$-simulates the abstract reduction system $(\mathcal{N},\rightarrow)$ is a little more subtle to establish and is a consequence of the following lemma proved first by J.-M. Hullot [26].

Lemma 3.10 *Let t_0 be a term and ρ be a normalized substitution. If $\rho(t_0)\rightarrow^R_{\omega,l\rightarrow r}t'_1$, then there exist substitutions σ and μ such that:*

(i) $t_0 \rightsquigarrow^R_{\omega,l\rightarrow r,\sigma} t_1$,
(ii) $\mu(t_1) = t'_1$, *μ is normalized,*
(iii) $\rho =_{\mathcal{V}ar(t_0)} \mu\sigma$ *(i.e. $\forall x \in \mathcal{V}ar(t_0), \rho(x) = \mu\sigma(x)$).*

This result is easily extended by induction on the number of steps to any rewriting derivation:

Proposition 3.11 *Let t_0 be a term and ρ be a normalized substitution such that:*

$$\rho(t_0)\rightarrow^R_{\omega_1,l_1\rightarrow r_1}\ldots\rightarrow^R_{\omega_n,l_n\rightarrow r_n}t'_n.$$

Then there exist substitutions $\sigma_i (i = 1,\ldots,n)$ and μ such that:

(i) $t_0 \rightsquigarrow^R_{\omega_1,l_1\rightarrow r_1,\sigma_1} t_1 \ldots \rightsquigarrow^R_{\omega_n,l_n\rightarrow r_n,\sigma_n} t_n$,
(ii) $\mu(t_n) = t'_n$, *μ is normalized,*
(iii) $\rho =_{\mathcal{V}ar(t_0)} \mu\sigma_n\ldots\sigma_1$ *(i.e. $\forall x \in \mathcal{V}ar(t_0), \rho(x) = \mu\sigma_n\ldots\sigma_1(x)$).*

Narrowing derivations thus $\Gamma_{\mathcal{N}}$-simulates rewriting derivations when strategies are not taken into account. In the more general case of strategic rewriting and narrowing, simulation results have only been stated for specific strategies such as innermost or outermost rewriting [22], and we address this problem here for priority rewriting [21].

Completeness of the narrowing process for solving equational goals has been largely studied in the litterature. A summary of results can be found in [1]. Termination of narrowing while preserving completeness has been less studied. Again a state of the art can be found in [1], as well as the definition of several classes of rewrite systems where narrowing has a finite search space, without requiring confluence of the rewrite system.

3.4 Simulation of IP-rewriting by IP-narrowing

Let us focus in this section on innermost priority rewriting and on the corresponding strategic narrowing relation. The idea is to capture any IP-rewriting $\alpha(t)\rightarrow^{IP}s$ by a constrained narrowing step $(t,c)\rightsquigarrow^{IP}(t',c')$ by setting adequately the constraints c, c'. For all normalized ground substitution α satisfying c' (and c), we should get an IP rewriting step on $\alpha(t)$ and a resulting term $\alpha(t')$. Constraints are first-order equational formulas, and the trivial constraint, always true, is denoted *id*.

Let us now analyze the construction of these constraints. In order to make possible the application of a rule $l\rightarrow r$ at some position ω in t, we first need to find

a mgu of $t_{|\omega}$ and l, say σ. Then to be IP-reducible by the rule $l \to r$ at position ω, a ground normalized instance $\alpha(t)$ must satisfy the following conditions:

- α is an instance of σ. Note that we can also consider a narrowing substitution σ of a term t at position ω with a rule $l \to r$, as the solved form of an equational constraint $t_{|\omega} =^? l$; in the following, we also denote by σ this solved form. Then α satisfies σ iff α is an instance of σ of the form $\alpha = \mu\sigma[\mathcal{V}ar(t) \cup \mathcal{V}ar(l)]$.
- There is no possible IP-rewriting at a position ω' suffix of ω (written $\omega' \geq \omega$):

$$\forall \omega' \geq \omega, \forall l' \to r' \in R, \forall \gamma, \alpha(t)_{|\omega'} \neq \gamma(l').$$

  Since α is normalized, ω' is a non-variable position in t. In other words α must satisfy a constraint c^{inn} which is a conjunction on all possible ω' and l' of disequality constraints:

$$c^{inn} = \bigwedge_{\omega' \geq \omega} \bigwedge_{l' \to r' \in R} \forall\, \mathcal{V}ar(l'), t_{|\omega'} \neq l'.$$

- There is no rule of higher priority to apply at position ω:

$$\forall l' \to r' \succ l \to r \in R, \forall \gamma, \alpha(t)_{|\omega} \neq \gamma(l').$$

  In other words α must satisfy a constraint c^{prio} which is a conjunction on all possible l' of disequality constraints:

$$c^{prio} = \bigwedge_{l' \to r' \succ l \to r \in R} \forall\, \mathcal{V}ar(l'), t_{|\omega} \neq l'.$$

A few renaming conditions are imposed in this narrowing process involving σ and l:

- variables of rules are kept disjoint from the variables of terms: $\mathcal{V}ar(t) \cup \mathcal{V}ar(\sigma(t))$ and $\mathcal{V}ar(l)$ are always disjoint sets;
- variables introduced by a mgu σ, that is $I(\sigma)$, are disjoint from $D(\sigma)$. This condition implies that unifiers are idempotent ($\sigma\sigma = \sigma$), so $\sigma(t') = \sigma(\sigma(t[r]_\omega)) = t'$;

We will need the three following lemmas, borrowed or adapted from [34].

Lemma 3.12 *If t is a term and σ a substitution, then $\mathcal{V}ar(\sigma(t)) = \mathcal{V}ar(t) \setminus D(\sigma) \cup I(\sigma)_{|\,\mathcal{V}ar(t)}$.*

It is useful to notive that, with the previous hypotheses, free variables of the constraints c^{inn} and c^{prio} belong to $\mathcal{V}ar(t)$. As a consequence of Lemma 3.12, if σ is the unifier used in the narrowing step, $\sigma(c^{inn}) = \bigwedge_{\omega' \geq \omega} \bigwedge_{l' \to r' in R} \forall\, \mathcal{V}ar(l'), \sigma(t)_{|\omega'} \neq l'$ and $\sigma(c^{prio}) = \bigwedge_{l' \to r' \succ l \to r \in R} \forall\, \mathcal{V}ar(l'), \sigma(t)_{|\omega} \neq l'$ have free variables in $\mathcal{V}ar(t) \setminus D(\sigma) \cup I(\sigma)_{|\,\mathcal{V}ar(t)}$.

Lemma 3.13 *Suppose we have substitutions σ, θ, θ' and sets A, B of variables such that $(B \setminus D(\sigma)) \cup I(\sigma) \subseteq A$. If $\theta = \theta'[A]$ then $\theta\sigma = \theta'\sigma[B]$.*

Lemma 3.14 *Let R be a rewrite system with priorities and suppose we have substitutions σ, θ, θ' and sets A, B of variables such that*

- $\theta_{|A}$ *is normalized*
- $\theta'\sigma = \theta[A]$
- $B \subseteq A \setminus D(\sigma)) \cup I(\sigma)_{|A}$.

Then $\theta'_{|B}$ is also normalized.

IP-narrowing is then defined on constrained terms, i.e. pairs (t, c) of terms and first-order equational constraints.

Definition 3.15 [Constrained IP-narrowing] Let t be a term and c a constraint. A constrained IP-narrowing step on (t, c) is defined if there exist a rule $l \to r$ of the system R, a non-variable position ω in t, a substitution σ solution of the constraint $t_{|\omega} =^? l$ and a constraint $c^{IP} = c^{inn} \wedge c^{prio}$ such that $\sigma \wedge c \wedge c^{IP}$ is satisfiable. Then $t' = \sigma(t[r]_\omega)$ and $c' = \sigma(c) \wedge \sigma(c^{IP})$. This is denoted by $(t, c) \leadsto^{IP}_{\omega, l \to r, \sigma} (t', c')$.

A *IP-narrowing* derivation starting from (t_0, c_0) is a sequence of constrained IP-narrowing steps.

Taking into account the restrictions put by constraints, we get the following result.

Lemma 3.16 *For any term $t \in \mathcal{T}(\mathcal{F}, \mathcal{Y})$ and rewrite system R with priorities, if $(t, c) \leadsto^{IP}_{\omega, l \to r, \sigma} (t', c')$ then for any ground normalized instance of σ, $\alpha = \mu\sigma$ such that μ satisfies c', $\alpha(t) \to^{IP}_{\omega, l \to r} \alpha(t')$.*

Proof Let us consider the term $s = \alpha(t)$. Its subterm $s|_\omega$ is an instance of l and ω is a position in t since α is normalized. Since α is an instance of σ, $\sigma(t) \to^R_{\omega, l \to r} t'$ and $\alpha(t) = \mu(\sigma(t)) \to^R_{\omega, l \to r} \mu(t') = \mu(\sigma(t[r]_\omega)) = \alpha(t')$. Moreover since μ satisfies $\sigma(c^{IP})$, α satisfies c^{IP}, so in particular of $\forall \mathcal{V}ar(l'), t_{|\omega'} \neq l'$ for all non-variable position $\omega' \geq \omega$ in t. So $\alpha(t)|_{\omega'}$ cannot be an instance of a left-hand side of rule in R, and any proper subterm of $s|_\omega$ is IP-irreducible. For the same reason, $s|_\omega$ cannot be reducible by a rule $l' \to r'$ having a higher priority than $l \to r$, since α is solution of $\forall \mathcal{V}ar(l'), t_{|\omega} \neq l'$. So the reduction step from $\alpha(t)$ to $\alpha(t')$ is innermost and respects priorities, and $s = \alpha(t)$ is IP-reducible to $\alpha(t')$. □

Consequently, an IP-narrowing derivation corresponds to a set of derivations in the abstract reduction system $(\mathcal{T}(\mathcal{F}), \to^{IP})$.

Proposition 3.17 *Let R be a rewrite system with priorities. For any term $t_0 \in \mathcal{T}(\mathcal{F}, \mathcal{Y})$, and any constraint c_0, if*

$$(t_0, c_0) \leadsto^{IP}_{\omega_1, l_1 \to r_1, \sigma_1} (t_1, c_1) \ldots \leadsto^{IP}_{\omega_n, l_n \to r_n, \sigma_n} (t_n, c_n)$$

then for any ground normalized instance of $\sigma = \sigma_n \ldots \sigma_1$, $\alpha = \mu\sigma$ such that μ satisfies c_n, we have

$$\alpha(t_0) \to^{IP}_{\omega_1, l_1 \to r_1} \ldots \to^{IP}_{\omega_n, l_n \to r_n} \alpha(t_n).$$

Proof By induction on the length of the IP-narrowing derivation. Lemma 3.16 provides the case $n = 1$. Then for any ground normalized substitution $\alpha = \mu\sigma$ such that μ satisfies $c_n = \sigma(c_0) \wedge \sigma(c_1^{IP}) \wedge \sigma(c_2^{IP}) \wedge \ldots \wedge \sigma(c_n^{IP})$, α may also be written $\alpha = \mu'\sigma_1$ with $\mu' = \mu\sigma_n \ldots \sigma_2$. Then μ' satisfies $c_1 = \sigma_1(c_0) \wedge \sigma_1(c_1^{IP})$, so $\alpha(t_0) \to^{IP}_{\omega_1, l_1 \to r_1} \alpha(t_1)$. By induction hypothesis, since $\alpha(t_1) = \mu\sigma_n \ldots \sigma_2(t_1)$ and μ satisfies c_n, $\alpha(t_1) \to^{IP}_{\omega_2, l_2 \to r_2} \ldots \to^{IP}_{\omega_n, l_n \to r_n} \alpha(t_n)$. □

Instantiating the initial constraint by the trivial constraint id which is always true, we get the following corollary.

Corollary 3.18 *Let R be a rewrite system with priorities. For any term $t_0 \in \mathcal{T}(\mathcal{F}, \mathcal{Y})$, if*

$$(t_0, id) \rightsquigarrow^{IP}_{\omega_1, l_1 \to r_1, \sigma_1} (t_1, c_1) \ldots \rightsquigarrow^{IP}_{\omega_n, l_n \to r_n, \sigma_n} (t_n, c_n)$$

then for any ground normalized instance of $\sigma = \sigma_n \ldots \sigma_1$, $\alpha = \mu\sigma$ such that μ satisfies c_n, we have

$$\alpha(t_0) \to^{IP}_{\omega_1, l_1 \to r_1} \ldots \to^{IP}_{\omega_n, l_n \to r_n} \alpha(t_n).$$

Conversely the following lifting lemma is inspired from [21] and [23].

Lemma 3.19 (IP-lifting Lemma) *Let R be a rewrite system with priorities. Let $s \in \mathcal{T}(\mathcal{F}, \mathcal{Y})$, α a substitution such that $\alpha(s)$ is IP-reducible at a non-variable position ω of s, and $V \subseteq \mathcal{Y}$ a set of variables such that $Var(s) \cup D(\alpha) \subseteq V$. If $\alpha(s) \to^{IP}_{\omega, l \to r} t'$, and if α satisfies a constraint c, whose free variables are included in V, then there exist a term $s' \in \mathcal{T}(\mathcal{F}, \mathcal{Y})$, a constraint c' and normalized substitutions μ, σ, such that:*

(i) $(s, c) \rightsquigarrow^{IP}_{\omega, l \to r, \sigma} (s', c')$

(ii) $\mu(s') = t'$

(iii) $\mu\sigma = \alpha[V]$

(iv) *μ satisfies c'.*

Proof We can always assume that $\mathcal{V}ar(l)$ and V are disjoint. We have $\alpha(s)_{|\omega} = \beta(l)$ for some β with $D(\beta) \subseteq \mathcal{V}ar(l)$. Since ω is a non-variable position in s, and since the domains of α and β are disjoint, $s_{|\omega}$ and l are unifiable by $\alpha \cup \beta = \gamma$. Let σ be an idempotent most general unifier of these two terms. Let $s' = \sigma(s[r]_{|\omega})$. Since σ is a mgu, there exists ρ such that $\rho\sigma = \alpha \cup \beta = \gamma$. Let $V' = (V \setminus D(\sigma)) \cup I(\sigma)$ and $\mu = \rho_{|V'}$. Clearly $D(\mu) \subseteq V'$. On the other hand, $\mathcal{V}ar(s') = \mathcal{V}ar(\sigma(s[r])) \subseteq \mathcal{V}ar(\sigma(s[l])) = \mathcal{V}ar(\sigma(s)) = \mathcal{V}ar(s) \setminus D(\sigma) \cup I(\sigma)_{|\mathcal{V}ar(s)}$ by Lemma 3.12, so $\mathcal{V}ar(s') \subseteq V'$. Thus we get $\mathcal{V}ar(s') \cup D(\mu) \subseteq V'$. Since $\mu = \rho[V']$, $\mu(s') = \rho(s') = \rho\sigma(s[r]) = \rho\sigma(s)[\rho\sigma(r)] = \alpha(s)[\beta(r)] = t'$.

Let us prove that $\mu\sigma = \alpha[V]$. Since $\mu = \rho[V']$ and $V' = (V \setminus D(\sigma)) \cup I(\sigma)$, we get by Lemma 3.13, $\mu\sigma = \rho\sigma[V]$, thus $\mu\sigma = \alpha[V]$.

Let us prove that μ satisfies the constraint $c' = \sigma(c) \wedge \sigma(c^{inn}) \wedge \sigma(c^{prio})$. Since the free variables of c, c^{inn}, c^{prio} are in V, the free variables of $\sigma(c) \wedge \sigma(c^{inn}) \wedge \sigma(c^{prio})$ are included in V'. Since $\mu\sigma = \alpha[V]$ and α satisfies c on V, then μ satisfies $\sigma(c)$. Since $\alpha(s) \to^{IP} t'$, α satisfies $c^{inn} \wedge c^{prio}$. Since $\mu\sigma = \alpha[V]$, then μ satisfies $\sigma(c^{inn}) \wedge \sigma(c^{prio})$.

Finally, due to (iii), if α is normalized, μ is also normalized by 3.14. □

This result can be extended by induction on the number of steps to rewriting derivations:

Proposition 3.20 *Let t_0 be a term, ρ be a normalized substitution, $V \subseteq \mathcal{Y}$ be a set of variables such that $Var(t_0) \cup D(\rho) \subseteq V$. If $\rho(t_0) \to^{IP}_{\omega_1,l_1\to r_1} t'_1 \ldots \to^{IP}_{\omega_n,l_n\to r_n} t'_n$, and if ρ satisfies a constraint c_0 whose free variables are included in V, there exist normalized substitutions $\sigma_i (i = 1,\ldots,n)$, constraints $c_i (i = 1,\ldots,n)$ and a ground normalized substitution μ such that:*

(i) $(t_0, c_0) \rightsquigarrow^{IP}_{\omega_1,l_1\to r_1,\sigma_1} (t_1, c_1) \ldots \rightsquigarrow^{IP}_{\omega_n,l_n\to r_n,\sigma_n} (t_n, c_n)$,

(ii) $\mu(t_n) = t'_n$,

(iii) $\rho = \mu\sigma_n \ldots \sigma_1[V]$,

(iv) *μ satisfies c_n.*

Proof The proof is by induction on the length of the derivation. For $n = 1$, it comes from Lemma 3.19. Let us assume the property for $n-1$ and consider $\rho(t_0) \to^{IP}_{\omega_1,l_1\to r_1} t'_1 \ldots \to^{IP}_{\omega_n,l_n\to r_n} t'_n$. According to Lemma 3.19, for any c_0 with free variables in V such that α satisfies c_0, there exists $(t_0, c_0) \rightsquigarrow^{IP}_{\omega_1,l_1\to r_1,\sigma_1} (t_1, c_1)$, $\beta(t_1) = t'_1$, $\beta\sigma_1 = \rho[V]$ and β satisfies c_1. Since β is normalized and satisfies c_1, and since $t'_1 = \beta(t_1) \to^{IP}_{\omega_2,l_2\to r_2} t'_2 \ldots \to^{IP}_{\omega_n,l_n\to r_n} t'_n$, by induction hypothesis, there exist normalized substitutions $\sigma_i (i = 2,\ldots,n)$, constraints $c_i (i = 2,\ldots,n)$ and a ground normalized substitution μ such that:

(i) $(t_1, c_1) \rightsquigarrow^{IP}_{\omega_2,l_2\to r_2,\sigma_2} (t_1, c_2) \ldots \rightsquigarrow^{IP}_{\omega_n,l_n\to r_n,\sigma_n} (t_n, c_n)$,

(ii) $\mu(t_n) = t'_n$,

(iii) $\beta = \mu\sigma_n \ldots \sigma_2[V']$,

(iv) μ satisfies c_n.

where V' contains $Var(t_1) \cup D(\rho)$. Since $\beta\sigma_1 = \rho[V]$, $\beta = \mu\sigma_n \ldots \sigma_2[V']$ and $V \setminus D(\sigma_1) \cup I(\sigma_1) \subseteq V'$, according to Lemma 3.13, we get $\rho = \beta\sigma_1 = \mu\sigma_n \ldots \sigma_2\sigma_1[V]$.□

Instantiating the initial constraint by the trivial constraint id which is always true, we get the following corollary.

Corollary 3.21 *Let t_0 be a term, ρ be a normalized substitution, $V \subseteq \mathcal{Y}$ be a set of variables such that $Var(t_0) \cup D(\rho) \subseteq V$. If $\rho(t_0) \to^{IP}_{\omega_1,l_1\to r_1} t'_1 \ldots \to^{IP}_{\omega_n,l_n\to r_n} t'_n$, there exist normalized substitutions $\sigma_i (i = 1,\ldots,n)$, constraints $c_i (i = 1,\ldots,n)$ and a ground normalized substitution μ such that:*

(i) $(t_0, id) \rightsquigarrow^{IP}_{\omega_1,l_1\to r_1,\sigma_1} (t_1, c_1) \ldots \rightsquigarrow^{IP}_{\omega_n,l_n\to r_n,\sigma_n} (t_n, c_n)$,

(ii) $\mu(t_n) = t'_n$,

(iii) $\rho = \mu\sigma_n \ldots \sigma_1[V]$,

(iv) *μ satisfies c_n.*

Example 3.22 Let us assume the following rules ordered as follows (higher priority

is the smaller label):

$$
\begin{aligned}
&1.\ pckt(src, dst, estab) &&\rightarrow accept\\
&2.\ pckt(eth0, dst, new) &&\rightarrow accept\\
&3.\ pckt(ppp0, dst, new) &&\rightarrow drop\\
&4.\ pckt(10.1.1.1, ppp0, s) &&\rightarrow pckt(123.123.1.1, ppp0, s)\\
&5.\ pckt(10.1.1.2, ppp0, s) &&\rightarrow pckt(123.123.1.1, ppp0, s)
\end{aligned}
$$

Let us apply IP-narrowing to the term $t = pckt(x, y, new)$ and the constraint identity. Since *new* and *estab* are two different constants, rule 1 does not apply. The substitution $\sigma = (x = eth0 \wedge y = y' \wedge dst = y')$ is a mgu with rule 2. Here

$$
\begin{aligned}
c' = \textstyle\bigwedge_{l' \rightarrow r' \in R} &\forall\, \mathcal{V}ar(l'), new \neq l' \wedge\\
&\forall src, dst, pckt(etho, y', new) \neq pckt(src, dst, estab)
\end{aligned}
$$

that is equivalent to true. And therefore: $t \leadsto^{IP}_{\epsilon,2,\sigma} accept$.

Rule 3 also applies with the substitution $\sigma' = (x = ppp0 \wedge y = y'' \wedge dst = y'')$. Here the constraint is

$$
\begin{aligned}
c' = \textstyle\bigwedge_{l' \rightarrow r' \in R} &\forall\, \mathcal{V}ar(l'), new \neq l' \wedge\\
&\forall z, pckt(ppp0, y'', new) \neq pckt(eth0, z, new) \wedge\\
&\forall z', z'', pckt(ppp0, y'', new) \neq pckt(z', z'', estab)
\end{aligned}
$$

that is equivalent to true again. And thus $t \leadsto^{IP}_{\epsilon,3,\sigma'} drop$. IP-narrowing with rules 4 and 5 is similar.

We are now ready to apply these results on strategic rewriting and narrowing to rewriting-based policies.

4 Narrowing-based analysis of rewriting-based policies

In our framework, policy evaluation is performed by strategic rewriting of ground requests. Consequently, for the safe design of security policies, several properties have been identified [14,11].

Definition 4.1 [Termination, Consistency, Decision Completeness] [11] A security policy $\wp = (\mathcal{F}, D, R, Q, \zeta)$ is

- *terminating* if for every $q \in Q$, all derivations of source q in ζ are finite.
- *consistent* if for every query $q \in Q$, ζ applied to q returns at most one result: $\forall q \in Q$, the cardinality of $\zeta q \cap D$ is less than or equal to 1.
- *decision complete* if $\forall q \in Q$, $\exists d \in D$, such that $d \in \zeta q$.

To deal with policy analysis, we add now the possibility to express queries:

Definition 4.2 [Queries] Given a security policy $\wp = (\mathcal{F}, D, R, Q, \zeta)$, a query is a term of $\mathcal{T}(\mathcal{F}, \mathcal{Y})$ where $\mathcal{Y}$ is a set of query variables distinct from $\mathcal{X}$. A constrained query is a pair of a term of $\mathcal{T}(\mathcal{F}, \mathcal{Y})$ and a first-order equational constraint.

Based on the previous results of Section 3, we can now analyze the behavior of security policies by solving queries. The method relies on the construction of a *narrowing tree* for a given query: all possible (strategic) narrowing steps are applied to this term and recursively to the resulting ones, yielding a possibly infinite tree. The general intuitive idea is that the narrowing trees provide a good simulation of the rewriting derivation trees that correspond to requests evaluation of the policy. We make the assumption that the security policy is terminating, in order to consider normalized terms, substitutions and instantiations. There are quite powerful techniques to check termination such as recursive path orderings [13], semantic labeling [37], dependency pairs [3], etc. These and other techniques have been implemented by several tools that allow to verify termination for a large sets of rewrite systems: for example AProVE [20], TTT [25], and CiME [32], to cite a few. All these verification methods and tools check termination statically, which is very attractive for the specification of flexible policies.

Considering innermost termination in particular, specific methods have been given for instance in [2,19,23]. Techniques have also been studied for termination of outermost rewriting [23], for termination of lazy rewriting [18] or for termination of priority rewriting [35,21]. Moreover it is often possible to prove termination in general (i.e. for the `universal` strategy), which implies termination under any strategy.

Example 4.3 The policy from Example 2.2 is not decision complete: the request $pckt(123.123.1.1, ppp0, new)$ is not reducible in this system and thus cannot be evaluated into a decision. If we add to the system the rule

$$pckt(123.123.1.1, ppp0, new) \to accept,$$

the policy becomes decision complete. In this example, a careful exhaustive exploration of all expressible requests and their reduction is possible. However in general, there is no method yet to prove this property formally for IP-rewriting. A promising approach may be found in [17] or [22].

The termination property can be easily checked by analyzing the rules of the policy and giving a simplification ordering $>$ such that $l > r$ for any left- and right-hand sides l, r of rules in R. This guarantees that the evaluation of any request will always terminate on a resulting term.

Consistency is also easily checked on this example since the critical pairs between rule 1 and rules 4 and 5 are convergent. However, note that confluence of IP-rewriting is not needed in the following, but would also require further study.

Thanks to the theoretical results of Section 3 stated both for rewriting and IP-rewriting, the narrowing-based analysis can be performed for IP-rewriting strategies

as well as for universal ones. More generally, whenever simulation of strategic rewriting by strategic narrowing can be established, the approach developed below for policy analysis can be followed. In order to cover IP and universal strategies, we consider in the following a narrowing relation on constrained terms, denoted by $\rightsquigarrow$ for simplicity.

4.1 Reachability analysis

Narrowing has been studied in [33] to solve reachability problems in cryptographic protocols verification. The problem is to find all solutions of a reachability goal $s \rightarrow^* t$, i.e. substitutions σ such that $\sigma(s) \rightarrow^* \sigma(t)$ with the rewrite system R which is terminating but possibly non-confluent. Narrowing is complete w.r.t. normalized substitutions: for every normalized substitution ρ, a more general solution η can be found by narrowing. The interested reader can refer to [33] to find other completeness results for solving reachability goals in particular classes of rewrite systems or of goals. These results can be useful for policy analysis and bear some similarities with the results we present below in the context of policy analysis.

4.2 What-if analysis

Let us first consider queries of the form $q(x_1, \ldots, x_n)$ where q is a term in $\mathcal{T}(\mathcal{F}, \mathcal{Y})$ with variables $x_1, \ldots, x_n \in \mathcal{Y}$.

For a constrained term (u, c), we denote by $\langle u, c \rangle$ the set $\{\theta(u) \mid \theta \in \Phi, \theta \in Sol(c)\}$ where Φ is the set of all ground normalized instances of $\mathcal{T}(\mathcal{F}, \mathcal{Y})$ into $\mathcal{T}(\mathcal{F})$ and $Sol(c)$ the set of ground normalized solutions of the constraint c.

Proposition 4.4 *Let us consider the narrowing tree of the constrained query (q, id).*

(i) *For every node (u, c) such that $(q, id) \rightsquigarrow_\sigma^* (u, c)$, all requests $\langle \sigma(q), c \rangle$ evaluate to ground instances of u in $\langle u, c \rangle$, where σ is the composition of normalized narrowing substitutions used in the narrowing derivation.*

(ii) *For any request t reachable by the policy from a request $\alpha(q)$, where α is normalized, there exists a node (u, c) in the narrowing tree of (q, id), such that $(q, id) \rightsquigarrow_\sigma^* (u, c)$ and $t = \mu(u)$, with μ normalized and μ satifies c.*

Proof

(i) is a direct application of Proposition 3.9 and Corollary 3.18.

(ii) directly follows from Proposition 3.11 and Corollary 3.21.

□

Based on this result, we can make some what-if analysis, as shown on the simple firewall Example 2.2.

Example 4.5 Suppose we want to know which packets get accepted for a new connection. This amounts to solve the query $pckt(x, y, new)$. We get the following

narrowing substitutions with the respective rules:

$$\begin{array}{lll}(1)\ x = eth0 & & y = dst\\ (2)\ x = ppp0 & & y = dst\\ (3)\ x = 10.1.1.1 & & y = ppp0\ new = s\\ (4)\ x = 10.1.1.2 & & y = ppp0\ new = s\\ (5)\ x = 123.123.1.1 & & y = ppp0\end{array}$$

giving respectively $accept, drop$ and $pckt(123.123.1.1, ppp0, new)$ twice, which can be reduced (i.e. narrowed with the identity substitution) to $accept$. These terms are no more narrowable. So possible requests that give an $accept$ decision are

- ground instances of $pckt(eth0, dst, new)$
- and the requests

$pckt(123.123.1.1, ppp0, new)$, $pckt(10.1.1.1, ppp0, new)$, $pckt(10.1.1.2, ppp0, new)$.

4.3 Analyzing decisions

Another interest of the approach is to provide also a methodology for analyzing the decisions taken by the policy. Let us first state some terminology and definitions. In this whole section, we assume that the policy $\wp$ is terminating and decisison complete.

A *decision term* is a ground term of D in normal form. A *query pattern* is a term p with variables $x_1, \ldots, x_n \in \mathcal{Y}$, that we will write $p(x_1, \ldots, x_n)$ when we want the variables to be explicit.

Let us assume from now on that the set Q of requests is defined as the set of ground instances of a finite set of *query patterns*. They are supposed non unifiable, so that in this way, every request is an instance of one query pattern.

The idea is that the narrowing trees built from this finite set of query patterns provide a correct and complete schematization of the request evaluation: correct means that every narrowing derivation represents a set of requests evaluations; complete means that every request evaluation is an instance of a narrowing derivation in the narrowing tree of its query pattern.

Then we can state the following results.

Proposition 4.6 *(Correctness) For a given security policy $\wp$ with query patterns, let us consider the narrowing trees of the constrained query patterns (p, id).*
If $(p, id) \leadsto^*_\sigma (u, c)$, u is a decision term and c is satisfiable, where σ is the composition of normalized narrowing substitutions, then the set $\langle \sigma(p), c\rangle$ is a set of requests leading to the decision u.

Proof This is a consequence of Proposition 4.4(i). Since u is a decision term, it is ground, so $\langle u, c\rangle = \{u\}$ provided c is satisfiable. □

Proposition 4.7 *(Completeness) For a given security policy $\wp$ with query patterns,*

for any request q which is a ground normalized instance α of the query pattern p, there exists a narrowing derivation $(p, id) \leadsto^*_\sigma (u, c)$ with c satisfiable, such that the evaluation of $q = \alpha(p)$ gives an instance of u in $\langle u, c\rangle$.

Proof This is a consequence of Proposition 4.4(ii). □

Provided decisions are normalized terms (which is trivial when decisions are constants on which no rewrite rule applies), for a security policy $\wp$ with query patterns, in order to generate all possible requests leading to a given decision, we can collect all the narrowing branches leading to this decision (obviously a leaf since a decision is no more reducible nor narrowable) and consider the union of the composed substitutions along each branch.

Proposition 4.8 *For a given security policy $\wp$ with m query patterns, let us consider the narrowing trees of the constrained query patterns (p_i, id) for $i = 1, ..., m$. Let d be a decision that appears in one or more narrowing trees of the constrained query patterns, say $(p_1, id) \leadsto^*_{\sigma_1} (d, c_1)$,..., $(p_k, id) \leadsto^*_{\sigma_k} (d, c_k)$ where σ_1,...,σ_k are the composed (constrained) substitutions. Then the set of ground normalized requests that evaluate to this decision d is $\langle\sigma_1(p_1), c_1\rangle \cup \ldots \cup \langle\sigma_k(p_k), c_k\rangle$.*

Proof By correctness, every set $\langle\sigma_i(p_i), c_i\rangle$ is a set of requests leading to d. Conversely, any request is a ground normalized instance of a pattern, and by completeness, there exists a narrowing derivation leading to d. □

One can also prove that a given pattern is not reachable, by trying to unify this pattern with each node. If no node unifies with the pattern, the latter is unreachable. In our framework, this gives the following result:

Proposition 4.9 *If a decision d is not an instance of any term u in a node (u, c) in the narrowing trees, then d is not reachable.*

Proof Assume that there exists a request leading to d. Then it would be a ground normalized instance of some pattern p and there would exist a rewriting derivation $\alpha(p) \xrightarrow{*} d$, with α ground normalized, and then a node (u, c) such that $(p, id) \leadsto^*_\sigma (u, c)$ and $d = \mu(u)$, which contradicts the hypothesis. □

Let us come back to Example 2.2.

Example 4.10 Suppose we want to know which packets get accepted and which are dropped. This amounts to solve the query $pckt(x, y, z)$. We get the following narrowing substitutions with the respective rules:

$$
\begin{array}{llll}
(1)\ x = src & y = dst & z = estab \\
(2)\ x = eth0 & y = dst & z = new \\
(3)\ x = ppp0 & y = dst & z = new \\
(4)\ x = 10.1.1.1 & y = ppp0 & z = s \\
(5)\ x = 10.1.1.2 & y = ppp0 & z = s
\end{array}
$$

Narrowing with the two first rules transforms the query $pckt(x, y, z)$ into *accept*, while the third rule transforms the query into *drop*. Narrowing with the two last rules transforms the query $pckt(x, y, z)$ into $pckt(123.123.1.1, ppp0, s)$. Then in each case a narrowing step can be again applied with the first rule, getting *accept* with $z = estab$. The two composed substitutions are $(6)(x = 10.1.1.1, y = ppp0, z = estab)$ and $(7)(x = 10.1.1.2, y = ppp0, z = estab)$.

There is then one more branch in the narrowing tree leading to *accept* with the substitution $(8)(x = 123.123.1.1, y = ppp0, z = new)$. A complete set of queries leading to the decision *accept* is given by the set of substitutions $\{(1), (2), (6), (7), (8)\}$.

5 Conclusion

The general context of this research is the design of a foundational framework for the compositional design, maintenance and verification of security policies. The framework initially presented in [14,12] addresses the problem of authoring and analyzing access control policies in a modular way using techniques developed in the field of strategic rewriting. Rewrite rules transform input terms representing access requests into access decision terms. In order to tame the raw computational power of term rewriting and to enhance the agility of the policy specification language, strategies are used to explicitly control the rule application. In this approach, the correspondences between the properties of the rewrite system and the policy it implements are easy to understand, thus we are able to directly apply a rich corpus of existing proof techniques and tools. We have proposed here to use narrowing to contribute to improve the trust of a user or an administrator with respect to a policy, not only at the design stage, but also whenever a policy is updated. However, adapting new concepts such as strategic rewriting or strategic narrowing is yet an ongoing goal which this paper begins to contributes to. Future work concerns fine-tuned analysis tools for policies, especially to formalize and tackle the various strategies used by developers of policies.

References

[1] Alpuente, M., S. Escobar and J. Iborra, *Termination of narrowing revisited* (2008), preprint submitted to Theoretical Computer Science.

[2] Arts, T. and J. Giesl, *Proving innermost normalisation automatically*, in: *Proceedings 8th Conference on Rewriting Techniques and Applications, Sitges (Spain)*, Lecture Notes in Computer Science **1232** (1997), pp. 157–171.

[3] Arts, T. and J. Giesl, *Termination of term rewriting using dependency pairs*, Theoretical Computer Science **236** (2000), pp. 133–178.

[4] Baader, F. and T. Nipkow, "Term Rewriting and all That," Cambridge University Press, 1998.

[5] Balland, E., P. Brauner, R. Kopetz, P.-E. Moreau and A. Reilles, "Tom Manual," LORIA, Nancy (France), version 2.4 edition (2006).

[6] Balland, E., P. Brauner, R. Kopetz, P.-E. Moreau and A. Reilles, *Tom: Piggybacking rewriting on java*, in: F. Baader, editor, *RTA*, Lecture Notes in Computer Science **4533** (2007), pp. 36–47.

[7] Barker, S. and P. J. Stuckey, *Flexible access control policy specification with constraint logic programming*, ACM Trans. Inf. Syst. Secur. **6** (2003), pp. 501–546.

[8] Borovanský, P., C. Kirchner, H. Kirchner and C. Ringeissen, *Rewriting with strategies in* ELAN*: a functional semantics*, International Journal of Foundations of Computer Science **12** (2001), pp. 69–98.

[9] Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. L. Talcott, *Reflection, metalevel computation, and strategies*, in: M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. L. Talcott, editors, *All About Maude*, Lecture Notes in Computer Science **4350** (2007), pp. 419–458.

[10] de Oliveira, A. S., *Rewriting-based access control policies*, Electr. Notes Theor. Comput. Sci. **171** (2007), pp. 59–72.

[11] de Oliveira, A. S., "Réécriture et modularité pour les politiques de sécurité," Ph.D. thesis, UHP Nancy 1 (2008).

[12] de Oliveira, A. S., E. K. Wang, C. Kirchner and H. Kirchner, *Weaving rewrite-based access control policies*, in: *FMSE '07: Proceedings of the 2007 ACM workshop on Formal methods in security engineering* (2007), pp. 71–80.

[13] Dershowitz, N., *Termination of rewriting*, Journal of Symbolic Computation **3** (1987), pp. 69–116.

[14] Dougherty, D. J., C. Kirchner, H. Kirchner and A. S. de Oliveira, *Modular access control via strategic rewriting*, in: *Proceedings of 12th European Symposium On Research In Computer Security (ESORICS'07)*, Dresden, 2007, pp. 578–593.

[15] Escobar, S. and J. Meseguer, *Symbolic model checking of infinite-state systems using narrowing*, in: F. Baader, editor, *Term Rewriting and Applications. 18th International Conference, RTA 2007*, Lecture Notes in Computer Science **4533**, 2007.

[16] Fay, M., *First order unification in equational theories*, in: *Proceedings 4th Workshop on Automated Deduction, Austin (Tex., USA)*, 1979, pp. 161–167.

[17] Fissore, O., I. Gnaedig and H. Kirchner, *A proof of weak termination providing the right way to terminate*, in: *Proceedings of the First International Conference on Theoretical Aspects of Computing ICTAC'2004*, Lecture Notes in Computer Science **3407** (2004), pp. 356–371.

[18] Giesl, J., S. Swiderski, P. Schneider-Kamp and R. Thiemann, *Automated Termination Analysis for Haskell: From term rewriting to programming languages.*, in: *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, 2006, pp. 297–312.

[19] Giesl, J., R. Thiemann, P. Schneider-Kamp and S. Falke, *Improving dependency pairs*, in: *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Artificial Intelligence **2850** (2003), pp. 165–179.

[20] Giesl, J., R. Thiemann, P. Schneider-Kamp and S. Falke, *Automated termination proofs with AProVE*, in: V. van Oostrom, editor, *RTA*, Lecture Notes in Computer Science **3091** (2004), pp. 210–220.

[21] Gnaedig, I., *Termination of Priority Rewriting*, in: A. Dediu, M. Ionescu and C. Martin-Vide, editors, *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications*, Lecture Notes in Computer Science (2009), preliminary version published as HAL-INRIA Open Archive Number inria-00243131. Available at http://hal.inria.fr/inria-00243131/en/.

[22] Gnaedig, I. and H. Kirchner, *Narrowing, abstraction and constraints for proving properties of reduction relations*, in: H. Comon-Lundh, C. Kirchner and H. Kirchner, editors, *Rewriting, Computation and Proof. Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science **4600** (2007), pp. 44–67.

[23] Gnaedig, I. and H. Kirchner, *Termination of rewriting under strategies*, ACM Transactions on Computational Logic **10** (2009), 56 p. Preliminary version as HAL-INRIA Open Archive Number INRIA-00113156.

[24] Gouda, M. G. and A. X. Liu, *Structured firewall design*, Computer Networks **51** (2007), pp. 1106–1120.

[25] Hirokawa, N. and A. Middeldorp, *Tyrolean termination tool: Techniques and features*, Inf. Comput. **205** (2007), pp. 474–511.

[26] Hullot, J.-M., *Canonical forms and unification*, in: W. Bibel and R. Kowalski, editors, *Proceedings 5th International Conference on Automated Deduction, Les Arcs (France)*, Lecture Notes in Computer Science **87** (1980), pp. 318–334.

[27] Jajodia, S., P. Samarati, M. L. Sapino and V. S. Subrahmanian, *Flexible support for multiple access control policies.*, ACM Trans. Database Syst. **26** (2001), pp. 214–260.

[28] Kirchner, C., F. Kirchner and H. Kirchner, *Strategic computations and deductions*, in: *Festchrift in honor of Peter Andrews*, Studies in logic and the foundations of mathematics, Elsevier, 2008 .

[29] Kirchner, C. and H. Kirchner, *Rewriting, solving, proving*, A preliminary version of a book available at `www.loria.fr/~ckirchne/rsp.ps.gz` (1999).

[30] Kirchner, C., H. Kirchner and M. Vittek, *Designing constraint logic programming languages using computational systems*, in: P. Van Hentenryck and V. Saraswat, editors, *Principles and Practice of Constraint Programming. The Newport Papers.*, The MIT press, 1995 pp. 131–158.

[31] Klop, J. W., *Term Rewriting Systems*, , **1**, Oxford University Press, 1990 .

[32] Marché, C. and X. Urbain, *Modular and incremental proofs of ac-termination*, J. Symb. Comput. **38** (2004), pp. 873–897.

[33] Meseguer, J. and P. Thati, *Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols*, Higher-Order and Symbolic Computation **20** (2007), pp. 123–160.

[34] Middeldorp, A. and E. Hamoen, *Completeness results for basic narrowing*, Journal of Applicable Algebra in Engineering, Communication and Computing **5** (1994), pp. 213–253.

[35] Mohan, C. K., *Priority rewriting: Semantics, confluence, and conditionals*, in: N. Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications, Chapel Hill (N.C., USA)*, Lecture Notes in Computer Science **355** (1989), pp. 278–291.

[36] Visser, E., *Stratego: A language for program transformation based on rewriting strategies. System description of Stratego 0.5*, in: A. Middeldorp, editor, *Rewriting Techniques and Applications (RTA'01)*, Lecture Notes in Computer Science **2051** (2001), pp. 357–361.

[37] Zantema, H., *Termination of term rewriting by semantic labelling*, Fundam. Inform. **24** (1995), pp. 89–105.