Strong Temporal, weak Spatial Logic for Rule Based Filters

Roger Villemaire
Université du Québec à Montréal
villemaire.roger@uqam.ca

Sylvain Hallé
University of California, Santa Barbara
shalle@acm.org

***Abstract*—Rule-based filters are sequences of rules formed of a condition and a decision. Rules are applied sequentially up to the first fulfilled condition, whose matching decision determines the outcome. Such filters are particularly useful in network management, where they filter packets allowed to flow in or out of an interface. Properties of filters which either reveal or hint to misconfiguration (anomalies) have been largely studied in the network management community. We show that in fact such properties are of a spatial and temporal nature. Accordingly we introduce a spatio-temporal language appropriate for filter properties, use it to describe major filter anomalies and finally prove that verifying a property in this language can be done in time polynomial in the number of filter rules.**

***Keywords*-component; formatting; style; styling;**

I. Introduction

Computer networks is an area of computer science where temporal aspects occur in many different contexts. Since the flow of network packets is controlled by the use of *protocols*, which are often modeled by automata or transition systems, temporal logics and model-checking techniques have been largely applied to the verification of protocols.

Other temporal aspects have received much less attention. For instance, in order to work properly, network equipments must be properly configured by setting numerous parameters to appropriate values. In network management, one concentrates not on the verification of the various protocols, but on the verification of the parameters, stored in network devices in some configuration structure, which configure these protocols.

This paper considers one such configuration structure: *rule-based filters*. These filters are widely used in network devices called firewalls, which link different zones of a network and filter network *packets* flowing from one zone into another. Filters are configured through a sequence of rules. Each rule is formed of a condition, which describes to which packets it applies, and a decision telling if the packet should be forwarded or blocked. For each packet, rules are applied sequentially up to the first rule whose condition is satisfied.

Since a rule applies only to packets which did not match previous rules, the task of modifying an existing filter is error-prone. An *anomaly* is a property of rule-based filters, which either reveal or at least hint to misconfiguration. It is therefore crucial to be able to detect such situations; consequently, anomaly-detecting algorithms and tools have been developed in the past years by the network management community. The best known algorithms have been shown to run in time exponential in n, the number of rules in a filter.

In this paper, we show that anomalies are in fact spatio-temporal properties of rule-based filters (section IV-B); they can be defined in a logic describing a region (the condition) evolving in time (the sequence of rules). This formalism can express all anomaly types presented in the literature. Furthermore, we show that model-checking a property definable in our logic on some rule-based filter can be done in time *polynomial* in the number of rules. Hence, it is not more expensive that the existing algorithms (no lower bounds have been published for these algorithms). Moreover, while existing algorithms each account only for a small subset of anomalies, model checking in our logic covers all anomalies at once. Since our logic allows the precise formalization of a broad range of properties at no algorithmic cost, we claim that it is an appropriate formalism for describing and verifying rule-based filters' anomalies.

II. Filters in Firewalls

In a global network such as the Internet, the level of trust can vary greatly. For instance corporate intranets, usually allow the exchange of sensitive information between trusted participants. It is hence crucial to enforce some *security policy* between the outer Internet and a corporate intranet. Similarly, an intranet can be divided into *zones*, enforcing various security policies. For instance a machine in the engineering zone could have total access to its own zone while having limited access to the accounting department zone.

Security policies are usually enforced through firewalls. A *firewall* is a dedicate machine or a router, which links different zones and filters packets flowing from one zone into another. For instance, the main corporate firewall is located between the outer Internet and the corporate intranet, and will usually limit access from outside to inner machines. Such firewalls can also limit traffic from the intranet into the Internet, for instance blocking access to high bandwidth streams such as video, in order to control intranet congestion —or concentrate employees on their tasks.

In the policy management field, a policy usually consists of roles, events and decisions which structure access control

to some resources (see [1] for a survey). In this paper we restrict ourself to low-level policies defined by rule-based filters.

A. Firewall Filters

A firewall contains an *interface* on each zone it connects to. Filtering is assured by filters, two by interface, one for incoming traffic and one for outgoing traffic. Furthermore each filter is usually configured through a sequence of rules (called *access-lists* in Cisco terminology [2, Chapter 7]). Each rule is formed of a *condition* describing to which packets it applies and a decision (typically *deny/accept*) which determines if the packet is forwarded or blocked.

Even if a rule's condition could in theory test any part of the incoming packet, usually the condition considers only the following *fields*: source and destination IP addresses, protocol (such as TCP, UDP, ICMP) and source and destination ports (for TCP and UDP). Packet fields values are taken from a finite ordered set, which is usually the set of n-bit natural numbers for some fixed length n, with the usual ordering. Typical examples are IP addresses (32 bits) or port numbers (16 bits).

A rule's condition is a conjunction of constraints of the form $f \in [\ell, u]$, where f is a packet field and $[\ell, u]$ the interval $\{x; \ell \leq x \leq u\}$. Usually, in configuration files, these intervals are described in a more network-centric notation such as the IP address/mask pairs in *dotted decimal* format (four dot-separated 8-bit values), a mask being a binary number where all 0's appear before any 1 (the inverse of the subnet mask, in accordance with Cisco convention). A pair such as `192.168.32.0/0.0.7.255` hence represents the set of IP addresses which agree with `192.168.32.0` on the 0's of the mask `0.0.7.255`, which is the interval `[192.168.32.0,192.168.39.255]`.

A *rule-based filter* (for short a *filter*) is a list of such rules. Each packet going through the filter is tested on every condition up to the first satisfied one. The decision associated with this *first matching* rule determines if the packet is accepted or rejected (*last matching* rule is also possible, but this is irrelevant for the results of this paper). Figure 1 shows a schematic filter with 5 rules and a single field.

B. Anomalies

It has been noticed in the network management community that understanding a deployed firewall packet-filtering policy can be a daunting task. Since a rule applies only to packets which did not match previous rules, the task of modifying an existing filter is error-prone. *Anomalies* are properties of filters which either reveal or hint to a possible misconfiguration. In order to help detect misconfigurations, a taxonomy of anomalies has been introduced [3]–[8].

The simplest type of anomalies, which relates two rules, was introduced by [3], [4]. They noticed that frequently, filter misconfiguration is due to the fact that some packet

rule	condition	decision
1	$[0, 5]$	accept
2	$[2, 3]$	deny
3	$[3, 10]$	accept
4	$[2, 10]$	accept
5	$[3, 3]$	deny

Figure 1. A schematic filter

satisfies many rules' conditions. It is then not obvious for a firewall manager to determine which rule applies to which packet nor how to properly order the rules. For instance, a rule r_1 is *simply shadowed* (e.g. rule 2 in Fig. 1) if there is a rule r_2 preceding r_1 in the filter, such that all packets satisfying r_1's condition already satisfy r_2's. In such a case r_1 applies to no traffic and is therefore either misplaced or unneeded.

Correlation happens when two rules' conditions match some packet while the rules have distinct decisions (e.g. rules 2 and 3 in Fig. 1). In this case the filter is not necessarily misconfigured, but it could be useful to inform the network engineer that the second rule' decision will not apply to all packets satisfying its condition.

Generalization happens when the second rule matches more than the first, but has a different decision (rule 4 generalize rule 2 in Fig. 1). While generalization has legitimate uses, such as rejecting packets from some host and then accepting traffic from all remaining machines on its subnet, it could be useful here also to inform the network engineer that the rule will not apply to all packets satisfying its condition.

A rule is *simply redundant* if it is simply shadowed by a rule with the same decision (e.g. rule 5 in Fig. 1). In this case, the rule can be removed without changing the packets that are accepted.

Finally, a rule is *irrelevant* if it applies to no traffic going through this filter. This happens for instance when the filter is located in an inner firewall, behind some other firewalls which already filter traffic. In such a case, filter rules should only consider traffic that can reach the filter. This allows a clean division of a global security policy within the different firewall filters.

These anomalies where generalized by [5], [6] to consider many rules. For instance a rule is *shadowed* (rule 4 is shadowed by rules 2 and 3 in Fig. 1) if all packets satisfying the rule's condition already satisfies some previous rule's condition (different packets can now satisfy different rules' conditions). Finally [7], [8] defined a rule to be *redundant* if removing it does not change the packet which are accepted.

Besides anomalies, [9] considered other important properties of filters, such as verifying that some services are not accessible from the Internet. Typical examples are:

- "Can I telnet from the Internet into my intranet?" or

- "Which (destination IP address, port number) pairs on the intranet can be reached from the Internet?"

C. Anomaly Detection Algorithms

From the above descriptions, it becomes crucial to be able to detect such situations. In recent years, anomaly detection algorithms have been proposed to this end by the network management community.

The algorithms presented in [3], [4] can detect anomalies. Since these anomalies only relate pairs of rules, the algorithms just have to test a pair of intervals for inclusion or intersection, which can be done simply by comparing endpoints.

Algorithms for managing *generalized* redundancy and shadowing are presented in [5], [6]. Their method transforms the filter into disjoint rules, such that no packet satisfies the conditions of two different rules. This method increases the total number of rules and an $O(p^n)$ upper bound for the runtime is given, where p is the number of fields and n is the number of rules. Moreover, an experimental evaluation shows that the method works fairly well in practice.

Alternately, Binary Decision Diagrams are used in [10] to represent sets of packets and test for shadowing, generalization, correlation and redundancy. This approach has been evaluated purely experimentally. The sets of packets accepted by filters are compared in [11] using a tree structure which represent a spatial decomposition of conditions into non-overlapping rectangles (or their higher dimension counterparts). They consider anomalies such as simple shadowing, correlation, generalization and simple redundancy and developed a prototype tool to evaluate their method.

Special decision trees data structures are introduced in [7], [8] to represent filters. Algorithms to remove redundancies are presented with neither theoretical nor experimental evaluation. Finally, [9] proposes a tool which answers queries such as those listed at the end of Section II-B. The tool simply computes for all rules the set of packets matching this rule and satisfying the query.

III. A Logic for Rule Filters

Filter properties are based both on the sequential order of the rules (the temporal order in which the rules are applied) and rules' conditions (spatial regions of a finite discrete space).

It is therefore attractive to consider applying spatial and temporal logics to formalize filters' properties.

We will now recall some results on spatial and temporal logics and make the requirements for a logical description of filter properties more precise.

A. Spatial Reasoning

Qualitative spatial reasoning has been largely studied particularly in IA [12]–[14]. An influential formalism is the Region Connection Calculus (RCC) [15] which describes properties of regions of a topological space. Unfortunately, RCC turned out to be undecidable [16], [17]. A propositional fragment, RCC-8, built up from 8 RCC-definable binary relations on region variables was later shown to be decidable [18] and in fact NP-complete [19]. Finally adding Boolean operations on region variables to RCC-8 yields BRCC-8, which was shown to NP-complete (PSPACE-complete for Euclidean spaces) by [20].

In order to verify filters' properties, we face a quite simple spatial situation: there is a single region (the condition), which is a subset of a finite discrete space. But this region evolves from rule to rule, introducing two kinds of temporal aspects. First, there are temporal properties which consider the rules independently one of another. This is the case in a statement such as "there exists a rule with an accepting decision". Secondly, there are properties which relate rules' conditions one to another. Shadowing is such a property, since it compares the conditions of two rules by inclusion. This second kind of properties is in fact central: shadowing, correlation, generalization and redundancy are all of this type.

Therefore, while our spatial needs may seem minimal, the temporal evolution of regions is a key aspect of filter properties. This issue must therefore be properly addressed by any logic which would be appropriate to describe filter properties.

B. Spatio-Temporal Reasoning

Fortunately, spatial and temporal logics have been integrated, yielding *spatio-temporal logics*, able to describe both spatial and temporal aspects.

Particularly interesting for us is the approach of [21], [22]. They combined *Propositional Temporal Logic* (PTL) [23], [24] —a temporal logic with Next, Since and Until operators —which they interpret on $\langle \mathbf{N}, < \rangle$— with BRCC-8, a propositional spatial logic. Here, a BRCC-8 formula describes a relationship between regions (such as inclusion) at some moment in time. Applying PTL temporal operators to BRCC-8 formulas gives a spatio-temporal logic called ST_0.

But, [21], [22] also noticed that temporal considerations applies not only to relationships between regions expressed as BRCC-8 formulas, but also to the regions themselves. For instance, considering some region C at some instant of time, one would like to compare it with the same region at the next instant, which will be denoted by $\bigcirc(C)$. They therefore defined temporal operators on regions in the following way.

As we already said $\bigcirc(C)$ represents the region C at the next instant. Furthermore $\Box^+(C)$ represents the intersection of the values of C in the (strict) future, $\Diamond^+(C)$ the union of the values of C in the (strict) future. The Until operator is *point-wise*, since the region $C_1 \mathbf{U} C_2$ is formed of every point x which is in C_2 at some moment in time and was in C_1 at all moments (strictly) before.

[21], [22] further introduced the extensions ST_1 of ST_0, which allows applying $\bigcirc$ on region variables, and ST_2 which further allows Until, Next (and possibly their past counterparts) on regions.

Finally [21] showed that decidability is PSPACE-complete for ST_0, in EXPSPACE for ST_1 and ST_2 (in this last case under the hypothesis that every region can have only finitely many states).

IV. Logics for rule-based filters

Even if filters' properties are of a spatio-temporal nature, usual spatio-temporal notation doesn't allow us to speak easily of individual rules. In order to have more readable logical expressions, it thus seems better to revert to a first-order representation. Nevertheless we will show in section IV-E that the temporal region operators, both for future and past, are definable in the logic we will now introduce, retaining its spatio-temporal nature.

A. A first-order Rule Language

We consider a many sorted first-order language RL. Its sorts are R for the rules and $F_1, \ldots, F_d$ for the packet fields. We will use an exponent to denote the variables' sorts, such as $\forall_r^R$ and $\exists_r^R$ for quantification on the R sort. The sort of a free variable should be clear from the context. As usual, each variable is of a unique sort. RL also contains binary relations symbols $\leq$, one for each sort. The intended meaning being the sequential order on rules for sort R and the orders on the fields' elements for the other sorts. For all sorts, we will use $x < y$ to denote $x \leq y \wedge x \neq y$. Finally, in order to specify a rule's condition, RL contains functions symbols $f_l^{F_i}(r), f_u^{F_i}(r)$ of sorts F_i, $i = 1, \ldots, d$ (l for lower and u for upper), where r is of sort R. The intended meaning being that a rule r's condition is that the packet's fields must be in $[f_l^{F_1}(r), f_u^{F_1}(r)] \times \cdots \times [f_l^{F_d}(r), f_u^{F_d}(r)]$. Finally RL also contains a predicate $d(r)$ to represent the fact that the condition is accept (if true) or reject (if false).

We will call the quantifiers $\forall^R$ and $\exists^R$ *rule quantifiers*. Furthermore since often quantifiers are applied globally to all packet fields, we will use $\forall_p^F$ (called *global quantifiers*) as a short-hand for $\forall_{p_1}^{F_1} \cdots \forall_{p_d}^{F_d}$ (similarly for $\exists_p^F$).

B. Examples

The set of packets with fields $(p_1, \ldots, p_d)$ satisfying rule r's condition is definable by the following RL-formula:

$$C(r, p_1, \ldots, p_d) \equiv \bigwedge_{i=1}^{p} f_l^{F_i}(r) \leq p_i \leq f_u^{F_i}(r)$$

Set theoretic operations (intersection, union) and relations (inclusion, equality) on these sets are therefore also definable in RL. In order to simplify notation, we will use $C(r, -)$ to represent the set of packets satisfying r's condition. This will allow to concisely express useful concepts, such as for instance the fact that all packets satisfying the condition of rule r also satisfy the condition of rule r', which we will denote by $C(r, -) \subseteq C(r', -)$. This set inclusion is obviously expressible in RL, by:

$$\forall_{p_1}^{F_1} \cdots \forall_{p_d}^{F_d} (C(r, p_1, \ldots, p_d) \rightarrow C(r', p_1, \ldots, p_d))$$

Furthermore we will freely use unions such as $\bigcup_{r'<r}$ for the union on all rules r' occurring before r, $\bigcup_{r'<r}^{d}$ the union on all rules r' occurring before r and having decision d, or $\bigcup_{r<r'<r''}$ the union on all rules occurring between r and r''. Such unions of RL-definable sets are again RL-definable.

Finally, since the first rule whose condition matches a packet is responsible for the decision, we will denote by $Match(r, -)$ the set $C(r, -) \setminus \bigcup_{r'<r} C(r', -)$.

We can now define in RL all the anomalies of section II-B.

Properties of a rule r:

- simply shadowed: $\exists_{r'}^R r' < r \wedge C(r, -) \subseteq C(r', -)$,
- shadowed: $C(r, -) \subseteq \bigcup_{r'<r} C(r', -)$,
- simply redundant: $\exists_{r'}^R r' < r \wedge (d(r) \leftrightarrow d(r')) \wedge C(r, -) \subseteq C(r', -)$,
- redundant: $Match(r, -) \subseteq \bigcup_{r'>r}^{d(r)} (C(r', -) \setminus \bigcup_{r<r''<r'} C(r'', -))$,
- irrelevance: $C(r, -) \cap TTF = \emptyset$.

For irrelevance (apply to no traffic going through this firewall), the traffic through this firewall, formed of all packets which weren't rejected by upstream firewalls, must be expressed by an RL-formula TTF. Since every firewall condition is a Boolean combination of comparisons between a packet field and a constant, TTF will also have this form.

Pairs of rules r, r':

- correlation: $C(r, -) \cap C(r', -) \neq \emptyset \wedge (d(r) \not\leftrightarrow d(r'))$,
- generalization: $r < r' \wedge C(r, -) \subseteq C(r', -) \wedge (d(r) \not\leftrightarrow d(r'))$,

Properties of a packet p: (here we use p as a shorthand for packet's fields $p_1, \ldots, p_d$)

- $Accept(p)$: $\exists_r^R C(r, p) \wedge d(r) \wedge \forall_{r'}^R (r' < r \rightarrow \neg C(r', p))$,

In fact, a firewall can be seen as a device carrying out the model-checking of $Accept(p)$ on every incoming packet p. RL does therefore not only describes properties of filters but also the filter outcome.

Queries

- "Can some packet with source address outside my intranet (IN =`[132.208.0.0,132.208.255.255]`) reach Telnet port (23) of a machine on my intranet?":

$$\exists_p^F (p_{srcIP} \notin IN \wedge p_{destIP} \in IN \wedge p_{destPort} = 23 \wedge Accept(p))$$

(this is a Boolean query, returning either True or False).

- "Which destIP/port on my intranet IN can be reached from the Internet?"

$$\exists^{F_{srcIP}}_{p_{srcIP}} \exists^{F_{srcPort}}_{p_{srcPort}} (p_{srcIP} \notin IN \wedge p_{destIP} \in IN \wedge$$
$$Accept(p_{sourcIP}, p_{srcPort}, p_{destIP}, p_{destPort}))$$

(this returns the values of the free variables p_{destIP}, $p_{destPort}$).

Filter properties

- Do not allow spoofed packet (coming from outside but with an internal source IP address). Applies to a filter on an incoming queue of an outer interface.
  $\forall^F_p (p_{srcIP} \in IN \rightarrow \neg Accept(p))$.
- Incoming packets for SMTP port (25) should have the mail server ($mail$) as destination IP address.
  $\forall^F_p (p_{destPort} = 25 \wedge Accept(p) \rightarrow p_{destIP} = mail)$.

C. Model-checking

We consider in this section the model-checking problem of an RL-formula on a filter. In our complexity analysis we will consider RL-formulas containing only global and rule quantifiers. Since a field quantifier ($\forall^{F_i}$, $\exists^{F_i}$) can always be replaced by a global quantifier ($\forall^F$, $\exists^F$), this does not restrict generality.

The simplest model-checking algorithm would be to test every possible element for every quantifier. This gives a time complexity of $O(|D|^{q_f} \cdot n^{q_r})$, where $D = D_1 \times \cdots \times D_d$, q_f is the global quantifier rank (maximal nesting of global quantifiers) and q_r is rule quantifier rank.

The size of a filter is the total size of its rules. A rule being a pair of elements of D (the upper and lower bounds in the rule's condition) and the decision (which can be considered as a single bit), the factor $|D|$ makes the above algorithm exponential in the size of the filter. Indeed, with a single IP address field, this algorithm would test all 32 bits IP addresses.

Let us now show that in fact RL model-checking, for a fix packet format (i.e. a fix number of fields) and fix quantifiers ranks, can be done in time polynomial in the number of rules.

Theorem 1: Model-checking an RL-formula on an n rule filter can be done in time $O(2^{d \cdot q_f^2} \cdot (2n+1)^{d \cdot q_f} \cdot n^{q_r})$, where q_f is the global quantifier rank and q_r is the rule quantifier rank.

Proof: We test again possible values for every quantifier. For a rule quantifier, we test every possible rule, yielding a n^{q_r} factor. But for global quantifiers we will test only some of the possible values.

A rule condition is formed of the Cartesian product of d intervals. The endpoints of the n rules' conditions i-th intervals divide D_i, into at most $2n+1$ intervals. An Ehrenfeucht-Fraïssé game argument similar to the one used to show that two linear orders of cardinality at least 2^k satisfy the same first-order formulas of quantifier rank at most k [25, Theorem 3.6], shows that keeping at most 2^{q_f} many elements in these $(2n+1)$ intervals gives a structure which satisfies the same RL-formula in global quantifier rank q_f and rule quantifier rank q_r. Note that if there are less than 2^{q_f} many elements in the interval, we will pick all of them, otherwise the choice of any 2^{q_f} many elements will do.

Picking values of global quantifiers in this new structure yields a time complexity of $O(((2n+1) \cdot 2^{q_f})^{d \cdot q_f} \cdot n^{q_r}) = O(2^{d \cdot q_f^2} \cdot (2n+1)^{d \cdot q_f} \cdot n^{q_r})$. ■

We now have a time complexity polynomial in n but still exponential in terms of d, q_f and q_c. Let us now show that unless $P = NP$ there are no model-checking algorithm polynomial in both n and d.

Theorem 2: Model-checking an RL-formula is NP-hard and this even if the formula contains only two rule and one global quantifier.

Proof: We will show in fact that there is a polynomial reduction of SAT to the model-checking of an RL-formula with two rule quantifiers and one global quantifier.

We define a filter whose rules' decisions are all 'reject' except for the last rule which accepts any packet. Such a filter satisfies the RL-formula $\exists^F_p Accept(p)$ if there is a packet which satisfies no condition except the last one. Let us now define these conditions.

Take d to be the number of propositional variables $P_1, \ldots, P_d$ in a CNF. Take $D_i = \{0, 1\}$, $i = 1, \ldots, d$. For any clause define a rule with the following condition. If P_i appears in the clause, add $f_i \in [0]$ to the condition, if $\bar{P}_i$ appears in the clause, add $f_i \in [1]$ to the condition. Finally if neither P_i nor $\bar{P}_i$ appears in the clause, add $f_i \in [0, 1]$.

Now the clause is satisfied if and only if the rule's condition is false. Therefore the CNF is satisfiable if and only if some packet is accepted by the filter. ■

To show that the exponents q_f, q_r in an RL model-checking algorithm are unavoidable (unless P=NP), we show that RL model-checking is in fact PSPACE-complete and this even if only one of q_f or q_r is allowed to vary.

Theorem 3: Model-checking an RL-formula is PSPACE-complete. It is still PSPACE-hard if at least one of q_f, q_r is not fix.

Proof: Membership into PSPACE follows from the fact that in our algorithm, in order to test every possible element for every quantifier, it suffices to store, besides the instance itself, the current value for each quantifier.

To show PSPACE-completeness, we polynomially reduce Satisfiability of Quantified Boolean (QBF), which is a known PSPACE-complete problem [26], to model-checking RL.

We will do this reduction in two different ways. First by polynomially encoding a QBF into a filter with a single rule and a single field (with at least two elements) and no rule quantifier. Secondly by polynomially encoding a QBF into a

filter with at least two rules, a single field (with any number of elements) and no field quantifiers.

In the first case, consider a filter with a single rule and a single field F with at least two elements. The smallest field element s is clearly definable in RL. Now encode a QBF into an RL-formula by replacing a quantified propositional variable Q_P by the quantifier Q_u^F on a new F-sort variable u and P by $u = s$. Since there are at least two fields values, u can always be chosen to make $u = s$ either true or false. Therefore this RL-formula φ is equisatisfiable with the original QBF.

Similarly, in the second case, consider a filter with at least two rules, the first rule r_s is again definable in RL. Encode now a QBF into an RL-formula by replacing a quantified propositional variable Q_P by the quantifier Q_r^R on a new R-sort variable r and P by $r = r_s$. Since there are at least two rules, this RL-formula is again equisatisfiable with the original QBF. ∎

The fact that our algorithm is exponential in d, q_f and q_c is not so important in practice, since usual values of d are small, such as $d = 5$ and, as we saw in section IV-B, most properties have small quantifier ranks.

D. Restricted Rule Language

It is still possible to improve model-checking run-time by a more detailed analysis of the properties of section IV-B. In fact, in all the examples of that section, comparison (by $\leq$ or $=$) on a field sort always contains either $f_l^{F_i}(r)$ or $f_u^{F_i}(r)$ on one side. If we define the *restricted rule language* RRL to contain all RL formulas in which comparison by $\leq$ or $=$ on a field sort contain either $f_l^{F_i}(r)$ or $f_u^{F_i}(r)$ on one side, we can show the following.

Theorem 4: Model-checking an RRL-formula on an n rule filter can be done in time $O((2n+1)^{d \cdot q_f} \cdot n^{q_r})$, where q_f is the global quantifier rank and q_r is the rule quantifier rank.

Proof: We test again all possible values for rule quantifiers, yielding the n^{q_r} factor.

For global quantifiers, we consider again the endpoints of the n rules' conditions i-th intervals, which divides D_i, into at most $2n+1$ intervals. Now since comparison with a quantified field variable occurs only with these end-points, it is sufficient to pick a single value in each of these $2n+1$ intervals. This yields a time complexity of $O((2n+1)^{d \cdot q_f} \cdot n^{q_r})$. ∎

E. Temporal Region Operators in RF

We chose to use a first-order language to describe filter properties since it offers the convenience of being able to name rules and packets. Nevertheless we want to make clear that all region temporal operators are easily expressible in RL. This follows from the fact that they are all definable by set-theoretic operations on time indexed regions. For instance $\bigcirc(C(r'-))$ is easily specified in RL by expressing the fact that it is the condition of the nearest future rule. $\Box^+(C(r,-))$ is simply $\bigcap_{r'>r} C(r',-)$. Likewise $\Diamond^+(C)$ and the Until operator can be expressed from their definition as is also the case for past operators.

V. Conclusion

We showed that filter properties, such as anomalies, can be describe using spatial and temporal reasoning. We introduced a spatio-temporal language RL, used it to describe all usual filter anomalies and finally proved that verifying an RL property on a filter can be done in time polynomial in the number of rules. Finally we noted that a restricted form of this language is sufficient to express all stated anomalies, which allowed us to still tighten our polynomial bound.

References

[1] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *Journal of Network and Systems Management*, vol. 15, no. 4, pp. 447–480, 2007.

[2] J. Boney, *Cisco IOS in a Nutshell*. O'Reilly, December 2001.

[3] E. S. Al-Shaer and H. H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004, pp. 2605–2616.

[4] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 10, pp. 2069–2084, 2005.

[5] F. Cuppens, N. Cuppens, and J. Garca-Alfaro, "Misconfiguration management of network security components," in *Proceedings of the 7th International Symposium on System and Information Security (SSI 2005)*. Sao Paulo, Brazil: ITA Corporate, 1 - 10, November 2005, p. (electronic medium).

[6] J. G. Alfaro, N. Boulahia-Cuppens, and F. Cuppens, "Complete analysis of configuration rules to guarantee reliable network security policies," *International Journal of Information Security*, vol. 7, no. 2, pp. 103–122, 2008.

[7] A. Liu and M. Gouda, "Complete redundancy detection in firewalls," in *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, ser. Lecture Notes in Computer Science, vol. 3654. Springer, 2005.

[8] M. G. Gouda and A. X. Liu, "Structured firewall design," *Computer Networks*, vol. 51, no. 4, pp. 1106–1120, 2007.

[9] A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *SP '00: Proceedings of the IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 177–187.

[10] 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*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 199–213.

[11] Y. Yin, R. Bhuvaneswaran, Y. Katayama, and N. Takahashi, "Analysis methods of firewall policies by using spatial relationships between filters," in *International Conference on Signal Processing, Communications and Networking (ICSCN '07)*, 2007, pp. 348–354.

[12] L. Vieu, "Spatial Representation and Reasoning in Artificial Intelligence," in *Spatial and Temporal Reasoning*, O. Stock, Ed. http://www.wkap.nl/: Kluwer, 1997, pp. 3–41. [Online]. Available: ftp://ftp.irit.fr/IRIT/LILAC/V-SRR97.pdf

[13] R. Casati and A. C. Varzi, *Parts and Places: The Structures of Spatial Representation*. MIT Press, 1999.

[14] A. G. Cohn and S. M. Hazarika, "Qualitative spatial representation and reasoning: An overview," *Fundamenta Informaticae*, vol. 46, no. 1-2, pp. 1–29, 2001.

[15] D. A. Randell, Z. Cui, and A. G. Cohn, "A spatial logic based on regions and connection," in *KR'92: 3rd International Conference on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992, pp. 165–176.

[16] N. M. Gotts, "Using the RCC formalism to describe the topology of spherical regions," Report 96.24, School of Computer Studies, University of Leeds, Tech. Rep., 1996.

[17] C. Dornheim, "Undecidability of plane polygonal mereotopology," in *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR-98)*. Morgan Kaufman, 1998, pp. 342–353.

[18] B. Bennett, "Spatial reasoning with propositional logics," in *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference (KR94)*. Morgan Kaufmann, 1994, pp. 51–62.

[19] J. Renz and B. Nebel, "On the complexity of qualitative spatial reasoning: a maximal tractable fragment of the region connection calculus," *Artificial Intelligence*, vol. 108, no. 1-2, pp. 69–123, 1999.

[20] F. Wolter and M. Zakharyaschev, "Spatial reasoning in RCC-8 with boolean region terms," in *Proceedings of the fourteenth European Conference on Artificial Intelligence, ECAI 2000, Berlin, Germany*, W. Horn, Ed. IOS Press, 2000, pp. 244–248.

[21] ——, "Spatio-temporal representation and reasoning based on RCC-8," in *Proceedings of the seventh Conference on Principles of Knowledge Representation and Reasoning, KR2000*. Morgan Kaufmann, 2000, pp. 3–14.

[22] ——, "Qualitative spatio-temporal representation and reasoning: a computational perspective," in *Exploring Artifitial Intelligence in the New Millenium*. Morgan Kaufmann, 2002, pp. 175–216.

[23] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems:Specification*. Springer-Verlag, 1992.

[24] ——, *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995.

[25] L. Libkin, *Elements of Finite Model Theory*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer Verlag, 2004.

[26] L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time: Preliminary report," in *STOC*. ACM, 1973, pp. 1–9.