(19) Europäisches Patentamt / European Patent Office / Office européen des brevets

(11) **EP 2 023 566 A1**

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
11.02.2009 Bulletin 2009/07

(51) Int Cl.:
H04L 29/06 *(2006.01)*

(21) Application number: **07114046.1**

(22) Date of filing: **08.08.2007**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC MT NL PL PT RO SE SI SK TR
Designated Extension States:
AL BA HR MK RS

(71) Applicants:
- **MITSUBISHI ELECTRIC CORPORATION**
Chiyoda-ku
Tokyo 100-8310 (JP)
Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI GB GR HU IE IS IT LI LT LU LV MC MT NL PL PT RO SE SI SK TR
- **Mitsubishi Electric R&D Centre Europe B.V.**
1119 NS Schiphol Rijk (NL)
Designated Contracting States:
FR

(72) Inventor: **Bouzida, Yacine**
35200 Rennes (FR)

(74) Representative: **Cabinet Plasseraud**
52, rue de la Victoire
75440 Paris Cedex 09 (FR)

Remarks:
Amended claims in accordance with Rule 137(2) EPC.

(54) **Online security rules conflict management**

(57) The present invention relates to a method of managing security rule conflicts online in an electronic device, such as a firewall. The electronic device applies a set *S* of security rules *R* to received data packets, and each rule comprising a list of conditional attributes *C* and at least one corresponding decision, the list of conditional attributes further comprising *m* individual conditional attributes C_n identified by corresponding intervals of values for setting a condition, where *n* is a variable identifying different conditional attributes in a rule. In case $m \geq 1$ then a conflict management tree is obtained, the tree comprising *m* levels of nodes, the nodes of different levels being connected to each other by arcs labeled by intervals or subintervals of the conditional attribute values and the nodes being characterized by a rule or rules fulfilling the condition set by the interval or subinterval labeling the arc leading to the corresponding node, the conflict management tree having leaves at the end of the arcs originating from the last level of nodes, each leaf being identified by a rule or rules that fulfill the condition set by the interval or subinterval labeling the arc leading to the current leaf and further being identified by the at least one corresponding decision. Next a new rule r is received and it is added to the set S of security rules. Then the conflict management tree is updated online based on the received new rule.

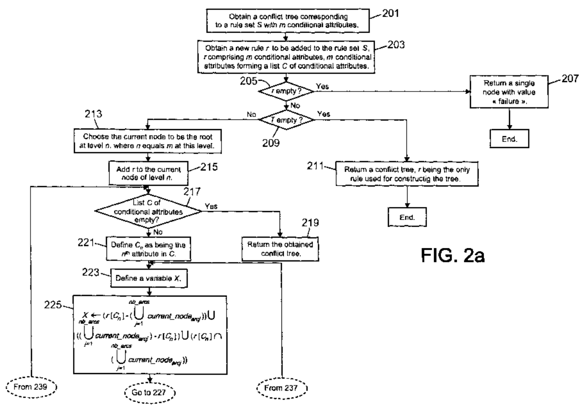


FIG. 2a

Printed by Jouve, 75001 PARIS (FR)

(Cont. next page)

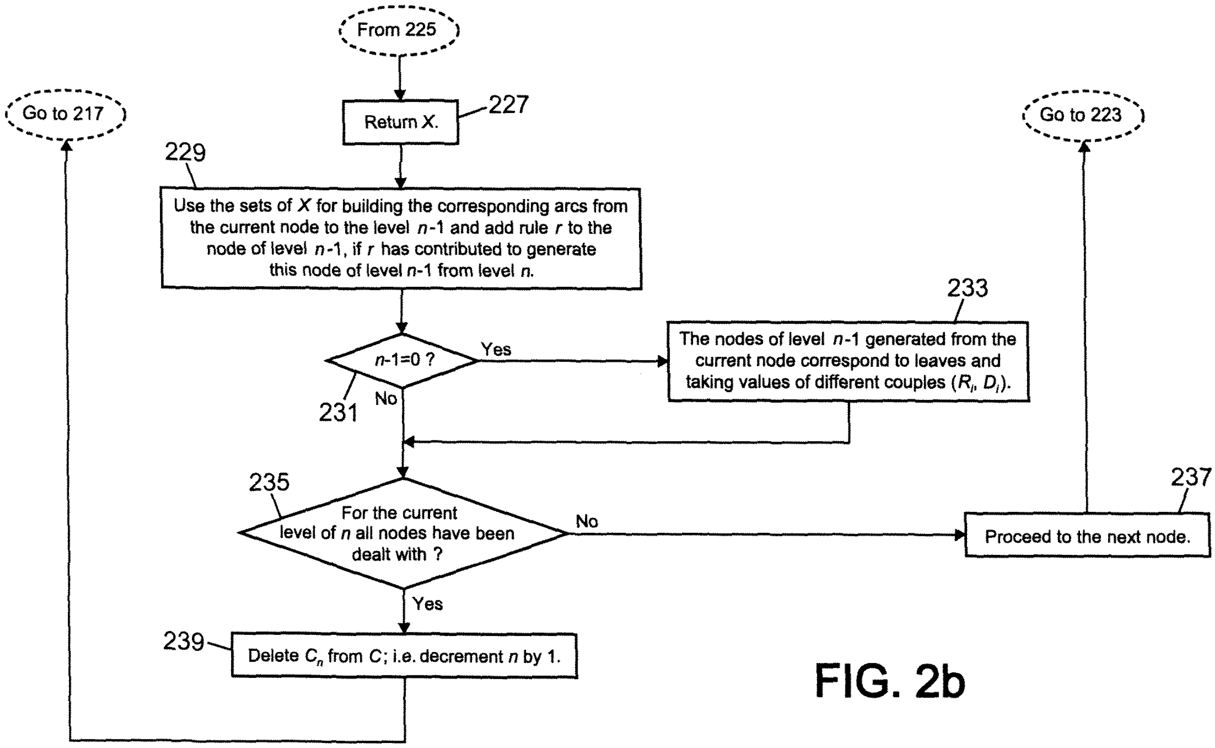

From 225
Return X.
227
Go to 217
Go to 223
229
Use the sets of X for building the corresponding arcs from the current node to the level n-1 and add rule r to the node of level n-1, if r has contributed to generate this node of level n-1 from level n.
233
n-1=0 ?
Yes
No
231
The nodes of level n-1 generated from the current node correspond to leaves and taking values of different couples (R_l, D_l).
235
For the current level of n all nodes have been dealt with ?
No
237
Proceed to the next node.
Yes
239
Delete C_n from C; i.e. decrement n by 1.

FIG. 2b

Description

TECHNICAL FIELD

[0001] The present invention relates to a conflict management method that can be used for detecting different anomalies present in a set of access control rules, when a new access control rule is added for updating an existing conflict management tree or building a new conflict management tree. Such access control rules can be used for instance in an electronic device for filtering incoming and outgoing traffic, in operating systems for managing file access, in attack signature databases for intrusion detection, in email applications for managing rules or in industrial mechanisms. The invention equally relates to such an electronic device and to a computer program arranged to implement the method.

BACKGROUND OF THE INVENTION

[0002] Current security mechanisms implemented in communication systems use a large number of techniques in order to filter packets in access control systems where many rules are specified for this task. Intrusion detection systems also use a set of rules to match different flows captured from the monitored network with the current known attacks that are described in a set of rules. The task of writing a rule; may it be for the access control goal or intrusion detection goal, is not a hard task. However, dealing with a large set of rules and discovering whether there are rules that are never tested or others that are redundant is not an easy task particularly when dealing with numerous rules written by successive administrators.

[0003] Security rules should be carefully written and well organized to answer the security policy specification. Since this is not an easy task that could be resolved by an operator, novel and fast automatic methods should be introduced to overcome this problem. Moreover, the administrator should be given exact information about the type of misconfiguration and the set of rules that conducted to such a misconfiguration.

[0004] Many current companies, governmental, academia and military agencies use a set of components to secure their information systems. Firewalls are the most used security equipment to filter and police the traffic traversing different zones of trust in an organization. Internet for example is considered as a zone with no trust and all the incoming and outgoing traffic to the Internet should be controlled with appropriate filtering rules to prevent attacks.

[0005] Although firewalls provide a powerful solution to filter traffic and control the flows traversing the different zones considered in an organization, their appropriate configuration according to the defined security policy remains a hard task. There are different reasons that make handling correct rules according to the target security policy difficult. First, the different firewalls of an organization may be installed in its different sub-organizations that are located in different places, countries and continents. Second, hundreds or thousands of rules are specified in the different firewalls or other security equipment pieces that may be written by different administrators along the time. Finally, updating the different rules according to new security policies by adding or removing rules renders the task of the administrator almost impractical. For all of these reasons, efficient techniques should be introduced to help administrators to verify, analyze and correct the different rules that are introduced.

[0006] There are various systems that try to solve the firewall misconfiguration problem. While these systems use different methods for detecting the different anomalies, they share approximately the same problems for detecting these anomalies.

[0007] A publication entitled "Discovery of Policy Anomalies in Distributed Firewalls" by Ehab S. Al-Shaer and Hazem H. Hamed, IEEE INFOCOM 2004, discloses a method for detecting anomalies, but this method does not detect all the anomalies that may exist when considering all the rules presented by an operator, because that detection approach is based on comparing the rules two by two.

[0008] The following illustrative example with only one conditional attribute dIP (destination IP) can be considered:

R_1: dIP $\in$ [40-90]$\rightarrow$ accept;
R_2: dIP $\in$ [80-120]$\rightarrow$ deny;
R_3: dIP $\in$ [50-110]$\rightarrow$ accept.

[0009] In the above rule set dIP denotes destination internet protocol (IP) address. In this case the proposed method does not detect the complete shadowing of rule R_3 because of the union of rules R_1 and R_2. A rule set contains a complete shadowing anomaly if there exists at least a rule that never applies because all the packets that this rule matches are already matched by a prior rule or a combination of rules that have higher priority in order.

[0010] A publication entitled "Detection and Removal of Firewall Misconfiguration" by F. Cuppens, N. Cuppens-Boulahia, and J. García-Alfaro, IASTED International Conference on Communication, Network and Information Security (CNIS 2005) discloses another solution that solves the above problem, but this method also has many shortcomings. An ambiguous definition of redundancy was introduced. As a matter of fact, if there are two rules that have the same attribute values, then the first one is considered as a redundant rule and so on if there are more than two identical rules. For this reason, the shadowing detection is considered before redundancy to restrict this shortcoming. However, if (1) one rule has higher priority in order than a set of one or more rules with lower priority with the same decision, and (2) all packets that the first rule matches are also matched by the combination of the other rules(s), and (3) the conditional attribute values of the first rule are strictly included in the conditional attribute values of the rules with lower priority, then the first rule is detected as redundant. In addition to this, if one rule set is considered and the rules are reordered then some anomalies might not be detected in the second rule set while they are detected in the first one and vice versa. The following rule set is an example that illustrates these two problems:

R_1: dIP $\in$ [20-90] $\rightarrow$ accept;
R_2: dIP $\in$ [40-120] $\rightarrow$ deny;
R_3: dIP $\in$ [10-55] $\rightarrow$ accept;
R_4: dIP $\in$ [30-110] $\rightarrow$ accept.

[0011] In this example, rule R_1 is detected as a redundant one due to the combination of rules R_3 and R_4. However, if the order of the rules R_1 and R_3 is altered as presented in the following, then R_1 is not detected as redundant even if it is actually redundant because if it is deleted from the list presented below, the result of filtering remains unchanged.

R_3: dIP $\in$ [10-55] $\rightarrow$ accept;
R_2: dIP $\in$ [40-120] $\rightarrow$ deny;
R_1: dIP $\in$ [20-90] $\rightarrow$ accept;
R_4: dIP $\in$ [30-110] $\rightarrow$ accept.

[0012] This situation is due to the ambiguity of that approach. That approach takes the different algorithms ahead rather than considering a real definition of redundancy. In addition to this, the execution order of the different rules is not kept according to the original rule set.

[0013] A special difficulty arises, if the administrator already has a set of security rules, but he has to add a new rule to the existing set. The task is even more difficult, if the rule set has to be updated online and conflicts detected online as well. Online updating means that rules can be added at any time instant. For instance if the algorithm described in the publication "Detection and Removal of Firewall Misconfiguration" is applied to online detection, the final result might not be the same as the result when applying this algorithm offline. This is due to the definition ambiguities. Moreover, different algorithms should be applied over all the rules for each new rule that is introduced online. However, this would be time consuming. Therefore, that method is not appropriate for online anomaly detection because of these two main drawbacks: time consumption and varying results. In addition, that method does not have memory keeping feature to save all the rules. Online conflict management is an innovative method that allows the detection of all possible anomalies step by step as long as the administrator introduces a new rule. Each time a new rule is introduced, the administrator is directly warned about the misconfiguration if there is any. Thus, when performing the misconfiguration detection online, the method is launched once a new rule is added.

[0014] Thus, there is a need for an improved online method for detecting and managing security rule conflicts.

SUMMARY OF THE INVENTION

[0015] According to a first aspect of the invention there is thus proposed an online conflict management method as recited in claim 1.

[0016] The proposed invention introduces a new method that is able to find all possible misconfigurations of different security rules that are specified by an administrator to filter inbound and outbound traffic traversing the considered filtering equipment or access rights when considering file access rights in operating system. There are many advantages of this mechanism when compared to other solutions. The proposed method is independent of rules reordering, i.e. rules priority. The method finds all possible anomalies; redundancy, shadowing, etc. Different anomalies can also be precisely presented to the administrator. Thus, all rules and their attribute range values that contributed to the anomaly can be precisely shown to the administrator for further investigation. Therefore, the administrator of the network element can be warned about all misconfigurations and it is up to him to choose the best rule set free of errors according to a priori defined policy. It is also possible to a priori tune the method so that the method gives automatically one possible rule set free of errors according to the administrator's choice.

[0017] The proposed method has also the advantage of "rules memory keeping". This method not only finds all misconfiguration anomalies and possibilities but also keeps at the same time the original rule set in its data structure as described later on. Furthermore, the method is invariant to updating of the rules. In contrast to other methods that give blindly the final rule set free of their corresponding anomalies, the proposed method keeps the original rule set while finding the different anomalies. The proposed method gives the same result irrespective of the order of the rules and all the anomalies are found while other methods depend on the last rules that are generated and the new rule that is added or omitted. Moreover, the proposed method makes it possible to update the rule set online. Thus, the administrator can be warned about all anomalies while he is introducing new rules in front of his console.

[0018] According to a second aspect of the invention there is further provided a computer program product comprising instructions for implementing the method according to the first aspect, when loaded and run on computer means of an electronic device that is able to manage the detected conflicts.

[0019] According to a third aspect of the invention there is provided a communication network element capable of managing conflicts as recited in claim 13.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] Other features and advantages of the invention will become apparent from the following description of non-limiting exemplary embodiments, with reference to the appended drawings, in which:

- Figure 1 is an example of a conflict management tree that can be updated in accordance with an embodiment of the invention;
- Figures 2a and 2b show a flow chart depicting a method in accordance with an embodiment of the present invention;
- Figure 3 shows a conflict management tree created online from only one security rule just after being introduced by a security rules administrator;
- Figures 4-6 show different stages of an updated conflict management tree when adding a new rule to an existing rule set;
- Figure 7 shows a conflict management tree free of rule anomalies;
- Figure 8 shows a part of another conflict management tree containing rule anomalies; and
- Figure 9 shows the conflict management tree of Figure 8 with one new added rule;
- Figure 10 shows the conflict management tree of Figure 9 illustrating the presence of redundant rules;
- Figures 11-12 show each a part of a conflict management tree of Figure 9 free of rule anomalies.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

[0021] Some embodiments of the invention will now be described in more detail with reference to the appended drawings. In the following description, the embodiments of the invention are described in the context of a communication system employing internet protocol (IP). However, the present invention is by no means limited to the use of IP.

[0022] The present invention may be implemented in a logical or a physical device inside filtering equipment. The only condition for the equipment is that it uses a set of rules that provides the corresponding actions or decisions to be applied when some conditions are satisfied. Examples of these devices are hardware firewalls, software firewalls (netfilter, ipfilter, etc.), intrusion detection/prevention systems (snort, dragon, etc.) that use a set of rules gathered in a signature database. Moreover, this technique may be generalized to other non security equipment such as quality of service (QoS) management tools, routers (routing table management, border gateway protocol aggregation, etc), optical network units (ONUs), optical line terminators (OLTs), gigabit ethernet passive optical networks (GePONs), messaging servers and client tools such as Microsoft™ Outlook when new rules are added, e.g. forwarding when receiving or sending new emails, etc. Moreover, the present invention is also useful for the new generation networks. In fact, this invention is recommended to be used for conflict configuration in session border controllers (SBCs) and in application layer gateways (ALGs) that have seen a great interest in the current voice over IP (VoIP) networks.

[0023] While the related work done in this field focused only on detecting some anomalies in a set of rules, the present invention applies not only to firewall misconconfigurations but also to intrusion detection rules such as snort's rules, files (or directory) access control in different Operating Systems (OSs) such as Windows, Linux, Unix, messaging servers and/or clients, industrial mechanisms and automation. In the following detailed description, the present invention is explained in more detail and its different procedures applied to a set of firewall rules since these rules illustrate efficiently the applicability of this invention.

[0024] Many current security equipment pieces which are either software or hardware devices, are mainly configured with a set of rules such as those introduced in Table 1. Generally, each rule provides a decision, such as accept, deny, alert, log, etc., according to a set of attribute values. In the case of firewall filtering function, these attributes may be related to the different IP packet header fields. However, this list of attributes is extensible to other features that are either related to application protocols such as the current VoIP protocols including session initiation protocol (SIP) or IP payload contents when dealing with rules in intrusion detection. For simplicity, in the following description some IP header fields such as protocol (P) (e.g. transmission control protocol, user datagram protocol, internet control message protocol, etc.), source IP address (sIP), destination IP address (dIP), source port (sP) and destination port (dP) are only considered.

The only condition that the different conditional attributes should fulfill is that their possible values belong to a finite set or a finite interval.

[0025] In the following, a grammar is provided that is used to describe the different rules where for each rule a set of conditions should be satisfied to accomplish the corresponding decision(s). Global syntax for a single rule can be defined in the following way:

R_i: <Conditions>: <Decisions>
<Conditions> → Condition ^ <Conditions> / Condition
<Decisions> → Decision <OP> <Decisions>/ Decision
<OP> → ∧/∨/...

[0026] In the syntax, OP denotes operand, ∧ denotes AND operation and ∨ denotes OR operation. The *Condition* terminal may be any condition where an attribute is tested whether it belongs to a set or not. This set may correspond to a single set, an intersection of many sets, a union of different sets and/or any other operation sets. The *Decision* terminal corresponds to any action that an element managing rules, such as a firewall, may apply such as *deny, accept, log,* etc. This set may contain discrete values, such as "tcp", "udp", "a", "b", "left" "right", "up", "down", "1", "2", "✸ ", "✱ ", "Paris" "London", "read", "write", "execute", etc., and/or continuous values such as intervals [-30; 5000], [20.36; 562], etc. However, these sets or intervals should be finite.

Table 1: Filtering rule set example with 5 conditional attributes.

Priority order	**Protocol (P)**	**Source IP (sIP)**	**Source Port (sP)**	**Destination IP (dIP)**	**Destination Port (dP)**	**Decision**
R_1	any	any	any	x.x.x.[20 - 90]	any	accept
R_2	any	any	any	x.x.x.[40 - 120]	any	deny
R_3	any	any	any	x.x.x.[1 - 55]	any	accept
R_4	any	any	any	x.x.x.[10- 80]	any	deny
R_5	any	any	any	x.x.x.[30 - 110]	any	accept
R_6	any	any	any	any	any	deny

[0027] From the global syntax presented above, the filtering rule syntax for the example of Table1 would be:

<Conditions> → <P> ∧ <SIP> ∧ <SP>∧ <DIP> ∧ <DP>
<Decision> → accept / deny.

[0028] The notation ∧ means logical AND operation. In Table 1, only five conditional attributes, or attributes in short, are present, namely P, sIP, sP, dIP and dP.

[0029] For the purpose of illustrating the online rules updating method, the different rules and the corresponding decisions are presented as a form of a conflict management tree. Figure 1 shows an example of such a tree. As can be seen from this figure, the tree contains three components: nodes, arcs and leaves. First, each node corresponds to a conditional attribute, for instance sIP is one conditional attribute in this example. Second, each arc is labeled with a conditional attribute value or a range of values. Finally, leaves correspond to couples, i.e. rule number and its corresponding decision. The leaves can be considered as special nodes. The different rules may be read by traversing the tree from up to down based on their conditional attribute values and the node values until one leaf is reached. Once a leaf is reached, two cases are possible. There is only one couple present, then the corresponding rule is free of anomalies according to the values of the arcs traversed from the root to the leaves of the conflict tree or there are at least two couples, which is an indication of a rule anomaly or conflict. In this latter case, there is a conflict between the different rules present in the tested leaf.

[0030] Next the method of updating rules online is described with reference to the flow chart of Figure 2. In step 201 a conflict management tree *T* is defined corresponding to a rule set *S*. The rule set contains different rules *R*, each having *m* conditional attributes. The tree *T* corresponds to a rule set to which an algorithm for constructing a conflict tree is applied. In step 203 a new rule *r* is provided by an administrator or any other agent such as a user for file access control and the proposed algorithm should then detect all the different anomalies that are present in the new rule set after adding the new rule *r.* This means that a new conflict tree should be constructed containing the preceding set of rules and the new added rule *r.* The new rule *r* to be added comprises m conditional attributes, where *m* is a positive integer.

[0031] The new rule *r* is tested in step 205. This means that it is determined whether *r* actually contains any conditional attributes, i.e. *r* is not empty. If *r* does not contain any conditional attributes, then in step 207 a single node is returned taking a value "failure". After this the procedure can be terminated.

[0032] On the other hand, if in step 205 it is determined that *r* indeed contains conditional attributes, then in step 209 it is determined whether *T* is empty, i.e. there is no existing conflict management tree meaning that the added rule *r* corresponds to the first rule introduced for example by the administrator. If *T* is empty, then in step 211 a new conflict management tree is created. The created tree contains as many nodes as there are conditional attributes, each node being identified with a corresponding conditional attribute. Furthermore, the different nodes are connected to each other through arcs which are labeled with corresponding conditional attribute values. At the end of the last conditional attribute value, i.e. *n*=1, there is a leaf, which can be considered to a special node. The leaf is identified with the rule *r* and its corresponding decision(s). After this step the procedure can be terminated.

[0033] Figure 3 shows an example of a constructed conflict management tree, when the following new rule is added and there is no existing tree: $P \in any \wedge sIP \in any \wedge sP \in any \wedge dIP \in x.x.x.[1\text{-}120] \wedge dP \in [21\text{-}80] \rightarrow accept$. As can be seen from the figure, there are five nodes identified by the corresponding conditional attributes and the rule *r* as there are five conditional attributes in the given rule *r.* Furthermore, the nodes are linked together by arcs, each arc being labeled or identified with the corresponding conditional attribute values. After the last node there is the leaf identified with the decision "accept".

[0034] However, if in step 209 it is determined that *T* is not empty, i.e. there is an existing conflict management tree, then in step 213 the current node is defined to be the root node at level *n,* i.e. the first node of the existing conflict management tree. *n* is a variable and it denotes the different node levels. At this level *n* equals *m.* In other words, when considering the given rule set *S*, the node corresponding to the first conditional attribute value can be considered as a root node. Next in step 215 the rule *r* is added to the current node of level *n*. Each node level corresponds to one conditional attribute. Thus, there are *m* different levels of nodes and after the last level there are the leaves corresponding to level 0. It is to be noted that each node level may have several parallel children nodes. A child node is defined to be a node that is connected to a father node or current node with an arc.

[0035] Then in step 217 the list C of conditional attributes is tested. In other words it is determined whether the list *C* of conditional attributes is empty. If this is the case, then in step 219 the obtained conflict tree is returned and the procedure terminates. If the list of conditional attributes *C* is not empty, then the procedure continues in step 221 by defining C_n as being the n^{th} attribute in *C*. Next in step 223 a variable *X* is defined.

[0036] In step 225 this variable is given the value of:

$$\left(r[C_n]-\left(\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj}\right)\right)\cup\left(\left(\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj}\right)-r[C_n]\right)\cup \left(r[C_n]\cap\left(\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj}\right)\right)$$

i.e. the union of the following sets:

$$r[C_n]-\left(\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj}\right), \qquad \left(\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj}\right)-r[C_n] \quad \text{and}$$

$r[C_n]\cap\left(\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj}\right)$, where $r[C_n]$ corresponds to the values of the conditional attribute of level n of the new rule *r.* In the example given in Figure 3 $r[C_2]= r[dIP]=[1\text{-}120]$. $\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj}$ denotes the union operation of all the values going from the current processed node of level *n* to the node of level *n*-1. For instance if we consider Figure 1 and the attribute of level 1 (i.e. attribute dP) and if we consider the second node at that level, i.e. the one where R_1, R_2 and R_3 are present, then the result is $(\bigcup_{j=1}^{nb\_arcs} current\_node_{arcj})$ = [10-20] ∪ [21-25] ∪ [26-80] ∪ [81-65535] = [10-65535]. Then in step 227 the current value of *X* is returned. The value *X* is comprised of the sets as defined in step 225.

[0037] Then in step 229, the sets as described in step 225 are used to build arcs from the current node to new nodes of level *n*-1 so that these arcs are labeled with these different sets. Thus, there are as many arcs going from the current node to the level *n*-1 nodes as there are number of sets. The nodes of level *n*-1 nodes are further identified with corresponding rules so that each node is identified only by the rules that fulfill the condition set by the corresponding set which labels the arc leading to the corresponding node. It is reminded that the sets correspond to the values of the conditional attributes or subranges of these values.

[0038] Next in step 231, the value of *n*-1 is tested by determining whether it is equal to 0. If this is the case, then the "node" of level *n*-1 created in step 229 (corresponding to a new arc) is returned as a leaf. Each returned leaf takes the values of corresponding couples (R_i, D_i), where R_i corresponds to the remaining rules in the current node and D_i corresponds to the corresponding decision(s), the subindex *i* is used to identify different rules and decisions. Each leaf is identified by a rule or rules that fulfill the conditions of the set labeling the arc leading to the current leaf. If there is no conflict in the current leaf, then only one couple is returned, whereas if there is a conflict, at least two couples are returned.

[0039] If the response to the test done in step 231 is negative then the procedure continues in step 235 by determining whether for the current conditional attribute C_n, or equally for the level *n*, all nodes have been dealt with. This is done by checking all the nodes of the previous level, i.e. level *n*+1, and verifying that all the sets have been dealt with, since there is always a node corresponding to a set. The set was defined in step 225, i.e. sets for obtaining the union. This can equally be expressed as checking that all the arcs from the previous level node(s) has/have been dealt with. If all the nodes have not been dealt with, then in step 237 the next node is considered and then the procedure continues in step 223. This loop continues as long as all the nodes have been dealt with.

[0040] On the other hand, if in step 235 it is determined that for the current level *n* all the nodes have been dealt with, then the procedure continues in step 239 by deleting the current C_n from *C*, i.e. *n* takes the value *n*-1. From step 239 the procedure goes directly back to step 217.

[0041] To illustrate the online conflict detection method as described with reference to the flow chart of Figures 2a and 2b, let us consider the conflict management tree shown in Figure 1. The following rule as a new rule is considered to be added:

R_4: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-200] ∧ dP ∈ [1-139] → accept.

[0042] It is to be noted that the first three conditional attributes are not changed compared to the first three conditional attributes of the existing rule set. Therefore, the execution of this algorithm is illustrated starting from the fourth attribute. The first three steps add only this last rule into the four nodes of levels (5, 4, 3 and 2) as can be seen in Figure 4, where the first three nodes are not only labeled by rules R_1, R_2, R_3, but also by R_4.

[0043] The 2nd (*n* = 2) conditional attribute is dIP. By performing steps 225 and 227, the arc labeled by a set [61-120] is split into two sets, namely into sets [61-69] and [70-120]. Steps 225 and 227 are called as a partitioning procedure in the following description. Figure 4 shows the state of the conflict management tree after this step has been performed.

[0044] The 1st (*n* = 1) conditional attribute is dP. By applying the partitioning procedure at this stage splits the arcs going from the last two dIP nodes of the above tree into four and three arcs respectively as shown in Figure 5.

[0045] Finally, when *n*-1 = 0 there is no conditional attribute to test. All of the five conditional attributes are tested. Therefore, a leaf is returned for each arc as shown in Figure 6. Each of them contains the remaining couples of the rule numbers and their corresponding decision. If one single leaf contains more than one rule, then this is an indication of an anomaly or conflict. In the figures the leaves with dashed contour lines are an indication of an anomaly.

[0046] When trying to find out different anomalies that may be driven from the conflict tree as shown in Figure 6 and generated by the present online conflict detection method, then directly from the tree it can be noticed that there are seven partial shadowing anomalies if we count one partial shadowing for each leaf where there is more than one rule and nine partial shadowing if we consider each rule independently such as the second dashed leaf from the left where there are three rules then two of them with lower priority are shadowed. Partial shadowing can be defined in the following way. Let *S* be a set of filtering rules. Then *R* presents a partial shadowing if and only if there exists at least a part of a rule R_i in *S* that never applies. This is due to the fact that all the packets that may be matched by this rule part are previously matched by at least one part of previous rules having a higher priority. In Figure 6, seven partial shadowing anomalies are:

- 1st partial shadowing {R_1, R_2}: P ∈ any ∧ sIP∈ any ∧ sP∈ any ∧ dIP ∈ x.x.x.[1-30] ∧ dP ∈ [21-25] → {R_1: accept,

R_2: deny}. If the priority used is the first match, i.e. R_1 has the highest priority then the packets that are matched simultaneously by R_1 and R_2, as detected by the conflict tree, would take always the decision of R_1 (accept) instead of that of R_2 (deny) and vice versa. The same partial shadowing anomaly occurs for the following part of the rules R_1, R_2 and R_3:

- 2nd partial shadowing {R_1, R_2, R_3}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∧ dP ∈ [21-25] → {R_1: accept, R_2: deny, R_3: deny}. In this case, we can consider two partial shadowing anomalies. The first one is that of R_2 that is shadowed by R_1. The second is that of R_3.
- 3rd partial shadowing {R_1, R_3}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∧ dP ∈ [26-80] → {R_1: accept, R_3: deny}.
- 4th partial shadowing {R_1, R_3}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[61-69] ∧ dP ∈ [21-80] → {R_1: accept, R_3: deny}.
- 5th partial shadowing {R_1, R_3, R_4}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [21-80] → {R_1: accept, R_3: deny, R_4: accept}.
- 6th partial shadowing {R_3, R_4}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [81-139] → {R_3: accept, R_4: accept }.
- 7th partial shadowing {R_3, R_4}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [21-139] → {R_3: deny, R_4: accept}.

[0047] There are two solutions proposed here to be provided to the administrator. The first consists in informing him about the different anomalies. This is done by drawing the tree as in Figure 6 or giving him the different paths from the root of the conflict tree to the leaves or by only giving him the partial shadowing anomalies as explained above.

[0048] The second solution consists in taking an automatic decision and the administrator is provided with the appropriate rules free of anomalies by deleting these anomalies automatically. The second technique is especially useful for novice administrators or users that perhaps do not have good knowledge about access control rules, file access rights or from any other field where this invention can be applied.

[0049] In accordance with the second solution the result is automated by providing a set of rules free of anomalies. In one aspect this solution involves considering the priority ordering of the rules. Therefore, if a rule R_i is first in order because of a higher priority than rule R_j, then R_i is taken as the rule for the path corresponding to the conflict. Therefore, this solution maintains the same matching order as the matching order of the first rule set provided by the administrator. In addition to this, the different attribute values that never apply are deleted from the first rule set. This solution is considered as an option in addition to the administrator warning.

[0050] A rule set is free from anomalies if its corresponding conflict tree that is generated does not contain leaves with more than one couple, i.e. a rule and its corresponding decision(s). For instance, the tree shown in Figure 7 is free from all anomalies. It is generated by choosing in each leaf the rule with the highest priority. A direct decision can be taken without considering the administrator decision since the conflict is taken in favor of the rules with higher priority.

[0051] Therefore, the administrator (or user or any other agent) is provided automatically with the following set of rules that is free from anomalies:

R_1: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-120] ∧ dP ∈ [21-80] → accept;
$R_{2'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-60] ∧ dP ∈ [10-20] → deny;
$R_{31'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-120] ∧ dP ∈ [81-65535] → deny;
$R_{33'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [21-65535] → deny; and
$R_{4'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-200] ∧ dP ∈ [1-20] → accept.

[0052] The rules generated from the conflict management tree are totally disjoint and are free from anomalies. Since they are disjoint then the result of their execution is the same when changing the order of the different rules. In the example given above R_1 is left unchanged. In fact, it is generated from the conflict tree in the following way:

R_{11}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-30] ∧ dP ∈ [21-25] → accept;
R_{12}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-30] ∧ dP ∈ [26-80] → accept;
R_{13}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∧ dP ∈ [21-25] → accept;
R_{14}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∧ dP ∈ [26-80] → accept;
R_{15}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[61-69] ∧ dP ∈ [21-80] → accept; and
R_{16}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [21-80] → accept.

[0053] Then this rule set is grouped into the following rule set:

$R_{1(1-2)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-30] ∧ dP ∈ [21-25] ∪ [26-80] → accept;
$R_{1(3-4)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∧ dP ∈ [21-25] ∪ [26-80] → accept; and
$R_{1(5-6)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[61-69] ∪ x.x.x. [70-120] ∧ dP ∈ [21-80] → accept.

[0054] This in turn is grouped into one rule:

$R_{1(1-6)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-30] ∪ x.x.x.[31-60] ∪ x.x.x.[61-69] ∪ x.x.x. [70-120] ∧ dP ∈ [21-25] ∪ [26-80] ∪ [21-80] → accept. This in turn is equivalent to the original rule R_1:
$R_{1,1-6}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-120] ∧ dP ∈ [21-80] → accept.

[0055] The other three rules R_2, R_3 and R_4 are obtained from the conflict management tree of Figure 6 in the following way:

R_{21}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-30] ∧ dP ∈ [10-20] → deny;
R_{22}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∧ dP ∈ [10-20] → deny;
R_{31}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∧ dP ∈ [81-65535] → deny;
R_{32}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[61-69] ∧ dP ∈ [81-65535] → deny;
R_{33}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [81-139] → deny;
R_{34}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [140-65535] → deny;
R_{35}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [21-139] → deny;
R_{36}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [140-65535] → deny;
R_{41}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [1-20] → accept; and
R_{42}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [1-20] → accept.

[0056] The above rule set is equivalent to the following set:

$R_{2(1-2)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-30] ∪ x.x.x.[31-60] ∧ dP ∈ [10-20] → deny;
$R_{3(1-2)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-60] ∪ x.x.x.[61-69] ∧ dP ∈ [81-65535] → deny;
$R_{3(3-4)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [81-139] ∪ [140-65535] → deny;
$R_{3(5-6)}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [21-139] ∪ [140-65535] → deny; and
$R_{4'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∪ [121-200] ∧ dp ∈ [1-20] → accept.

[0057] This in turn is equivalent to the following rule set:

$R_{2'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-60] ∧ dP ∈ [10-20] → deny;
$R_{31'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-69] ∧ dP ∈ [81-65535]→ deny;
$R_{32'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-120] ∧ dP ∈ [81-65535] → deny;
$R_{33'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [21-65535] → deny; and
$R_{4'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-200] ∧ dP ∈ [1-20] → accept.

[0058] This in turn is equivalent to the following rule set that is free from anomalies:

R_1: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-120] ∧ dP ∈ [21-80] → accept;
$R_{2'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[1-60] ∧ dP ∈ [10-20] → deny;
$R_{31'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[31-120] ∧ dP ∈ [81-65535] → deny;
$R_{33'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[121-200] ∧ dP ∈ [21-65535] → deny; and
$R_{4'}$: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[70-200] ∧ dP ∈ [1-20] → accept.

[0059] To illustrate other capabilities of the present invention, let us consider the example presented in Table 1 that contains redundancy anomaly as defined in the publication entitled "Detection and Removal of Firewall Misconfiguration". In that document the redundancy was defined in the following way. Let *S* be a set of filtering rules. Then *S* contains a complete redundancy if and only if there exists at least one filtering rule R_i in *S*, such that the following conditions hold: (1) R_i is not shadowed by any other rule; (2) when removing R_i from *S*, the filtering result does not change. To check whether a rule R_i is completely redundant, it is sufficient to check each leaf, where this leaf is present, the presence of at least a rule with the same decision. For presentation conveniences, only the node level corresponding to the destination IP (dIP) conditional attribute is described next in more detail. Figure 8 illustrates this situation.
[0060] Let us consider a situation, where a new rule, namely rule R_{34} is added to the existing rule set between rules

R_3 and R_4: R_{34}: P ∈ any ∧ sIP ∈ any ∧ sP ∈ any ∧ dIP ∈ x.x.x.[30-130] ∧ dP ∈ any → deny. The administrator adds this rule between R_3 and R_4 as shown in Table 2 below so that the added rule has the fourth highest priority.

Table 2. New rule set with added rule R_{34}.

Priority order	Protocol (P)	Source IP (sIP)	Source Port (sP)	Destination IP (dIP)	Destination Port (dP)	Decision
R_1	any	Any	any	x.x.x.[20-90]	Any	accept
R_2	any	any	any	x.x.x.[40-120]	Any	deny
R_3	any	any	any	x.x.x.[1-55]	Any	accept
R_{34}	**any**	**any**	**any**	**x.x.x.[30-130]**	**Any**	**deny**
R_4	any	any	any	x.x.x.[10-80]	Any	deny
R_5	any	any	any	x.x.x.[30-110]	Any	accept
R_6	any	any	any	any	Any	deny

[0061] By using the method as described with reference to the flow chart of Figures 2a and 2b and by using the existing tree as shown in Figure 8, a new tree is obtained. Figure 9 shows the level corresponding to the dIP conditional attribute, when the new rule R_{34} has been added to the existing rule set. From the obtained conflict management tree different anomalies can now be detected.

For checking the redundancy as defined in the publication entitled "Detection and Removal of Firewall Misconfiguration", it is sufficient to look at different leaves in the conflict tree and check for each rule the presence of another rule with a lower priority and with the same decision as that of the tested rule. For instance, according to the conflict tree shown in Figure 9, rule R_1 is detected as redundant since it appears that in every leaf there is at least another rule with the same decision and a lower priority. In fact, R_3 is present with a lower priority and with the same decision in the first three leaves where R_1 shows its presence, and R_5 with a lower priority and with the same decision is present in the last three leaves where R_1 is present. In the third leaf where R_1 is present, the two rules R_3 and R_5 are both present. To detect a general redundancy despite the order of a rule, as for example R_1 as presented in Table 2, it is sufficient to check the presence of another rule with the same decision as the tested rule with a lower or a higher priority. With the same manner R_2 is detected as redundant. Figure 10 shows the redundant rules R_1 and R_2.

[0062] For the detection of shadowing anomaly as defined in the publication entitled "Detection and Removal of Firewall Misconfigurations", it is sufficient to keep only one rule in each leaf after applying the redundancy algorithm. A rule set *S* contains a complete shadowing anomaly if there exists at least a rule R_i in *S* that never applies because all the packets that this rule matches are already matched by a prior rule or a combination of rules that have higher priority in order. Then R_i is called a completely shadowed rule. For the complete shadowing detection of a rule R_i, by using the method proposed in the present invention, it is sufficient to check each leaf where this rule is present. So if in all of the leaves where it is present, it is partially shadowed by a finite and not empty set of rules that have a higher priority in order, then this rule is completely shadowed. Therefore, if any rule is not present in any leaf of the new generated conflict tree then it is considered as a shadowing anomaly. One option to do this involves deleting the rules with a lower priority in a leaf. Figure 11 presents the result of applying the shadowing algorithm after having applied the redundancy algorithm. From the conflict tree shown in Figure 11, the following set of rules, which is free of errors, is generated:

$R_{3'}$: dIP ∈ x.x.x.[1-55] → accept;
$R_{34'}$: dIP ∈ x.x.x.[91-130] → deny;
$R_{5'}$: dIP ∈ x.x.x.[56-90] → accept.

[0063] This set of rules is obtained by applying the redundancy and shadowing anomalies definitions as introduced in the publication entitled "Detection and Removal of Firewall Misconfiguration".

[0064] However, the following anomaly definition can also be used: *R* presents a partial shadowing if and only if there exists at least a part of a rule R_i in *S* that never applies. If this definition is used on the original conflict tree shown in Figure 9, then the corresponding conflict tree that is free of anomalies is shown in Figure 12. The result is obtained in Figure 12 by simply taking the first rule in each leaf and removing the other rules in the current leaf. This choice will not alter the execution and comparison ordering of the obtained rules by the corresponding equipment when compared to the execution of the first rule set as it is introduced initially.

[0065] From the conflict tree shown in Figure 12, the following rule set, which is free of errors, is generated. Only the dIP conditional attribute is considered for the matter of illustration:

$R_{1'}$: dIP ∈ x.x.x.[20-90] → accept;

$R_{2'}$: dIP ∈ x.x.x.[91-120] → deny;

$R_{3'}$: dIP ∈ x.x.x.[1-19] → accept;

$R_{34'}$: dIP ∈ x.x.x.[121-131] → deny.

[0066] It is to be noted that the rule set generated from the conflict tree in Figure 11 is different from that generated from the conflict tree of Figure 12. This is due to the definition of redundancy introduced in the publication entitled "Detection and Removal of Firewall Misconfiguration". In fact, for the first rule set the execution and comparison orders according to the original rule set are not maintained while this is not the case when using the option proposed in the present invention. The solution for obtaining the result as shown in Figure 12 may be considered better in certain instances, especially because it is quite simple to implement.

[0067] The administrator may choose appropriate measures to be taken from the conflict tree that is automatically generated from the algorithm of the flow chart of Figures 2a and 2b. By using the different rules in the leaves of the conflict tree, misconfigurations can be found out such as complete shadowing and complete redundancy. One can define these misconfigurations differently than as was defined above. Despite the definition of a rule misconfiguration, its detection depends directly on the different rules that are present in the different conflict tree leaves. A rule may be completely shadowed and completely redundant at the same time. Many other anomalies may be defined such as generalization, correlation, etc. All of these anomalies may be detected from the conflict tree and particularly from the leaves.

[0068] The rules generated from the conflict tree as shown in Figure 12 are totally disjoint and are free of anomalies. Since they are disjoint then the result of their execution is the same when changing the order of the different rules.

[0069] The invention equally relates to a computer program product that is able to implement any of the method steps of the embodiments of the invention when loaded and run on computer means of an electronic device.

[0070] The invention equally relates to an electronic device that is arranged to implement the method steps described above. The computer program can be arranged to be run by the electronic device.

[0071] Thus, the present invention provides a robust and concise method which is able to detect all possible anomalies in an optimum way, i.e. space and time consumptions are minimized. Furthermore the present invention enables updating the rule set and thereby the related conflict management tree online. Moreover, the embodiments of the present invention let the network administrator choose the appropriate combination of rules that is free of anomalies. The decision may be automatic or manual according to the preferences of the administrator.

[0072] While the invention has been illustrated and described in detail in the drawings and foregoing description, such illustration and description are to be considered illustrative or exemplary and not restrictive; the invention is not restricted to the disclosed embodiments.

[0073] Other variations to the disclosed embodiments can be understood and effected by those skilled in the art in practicing the claimed invention, from a study of the drawings, the disclosure and the appended claims. In the claims, the word "comprising" does not exclude other elements or steps, and the indefinite article "a" or "an" does not exclude a plurality. A single processor or other unit may fulfill the functions of several items recited in the claims. The mere fact that different features are recited in mutually different dependent claims does not indicate that a combination of these features cannot be advantageously used. Any reference signs in the claims should not be construed as limiting the scope of the invention.

Claims

1. An online method of managing security rule conflicts in an electronic device arranged to store a set *S* of security rules *R*, each rule comprising a list of conditional attributes *C* and at least one corresponding decision, the list of conditional attributes further comprising *m* individual conditional attributes C_n identified by corresponding intervals of values for setting a condition, where *n* is a variable identifying different conditional attributes in a rule, the method comprises the following steps performed by the electronic device:

- upon determining that $m \geq 1$, obtaining (201) a conflict management tree comprising *m* levels of nodes, the nodes of different levels being connected to each other by arcs labeled by intervals or subintervals of the conditional attribute values and the nodes being **characterized by** a rule or rules fulfilling the condition set by

the interval or subinterval labeling the arc leading to the corresponding node, the conflict management tree having leaves at the end of the arcs originating from the last level of nodes, each leaf being identified by a rule or rules that fulfill the condition set by the interval or subinterval labeling the arc leading to the current leaf and further being identified by the at least one corresponding decision;
- receiving (203) a new rule *r* to be added to the set *S* of security rules; and
- updating (229) online the conflict management tree based on the received new rule.

2. The method according to claim 1, wherein the updating comprises:

a) defining (223) a variable *X*;

b) *X* taking the value of $\left(r[C_n] - \left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj} \right) \right) \cup$

$$\left(\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj} \right) - r[C_n] \right) \cup \left(r[C_n] \cap \left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj} \right) \right),$$

where $\cup$ denotes mathematical operation union, $\cap$ denotes mathematical operation intersection, $current\_node_{arcj}$ denotes the intervals or subintervals of an arc *j* from the current node to the level n-1 node and where $\left(r[C_n] - \left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj} \right) \right)$, $\left(\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj} \right) - r[C_n] \right)$ and $\left(r[C_n] \cap \left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj} \right) \right)$ are sets of the union operation corresponding to the intervals or subintervals of the conditional attributes C_n;
c) using (229) the sets of the union operation for forming arcs to level *n*-1 nodes so that each set is used to label only one arc, wherein the level *n*-1 nodes are identified by the corresponding rules so that each node is identified only by the rules that fulfill the condition set by the corresponding set labeling the arc leading to the corresponding node.

3. The method according to claim 2, wherein the updating further comprises:

d) determining (231) whether *n*-1 equals to 0; and
e) in case *n*-1 equals to 0, then adding (233) leaves to the level *n*-1 of the current node, the leaves being identified by a rule or rules that fulfill(s) the conditions of the set labeling the arc leading to the corresponding leaf.

4. The method according to claim 3, wherein the updating further comprises:

f) in case n-1 does not equal to 0 or after having performed step e), then determining (235) whether all nodes of the current level n have been dealt with, and if this is not the case then repeating steps a) to d) until all the nodes of the current level n have been dealt with; and
g) decrementing (239) *n* by 1 and then repeating steps a) to f).

5. The method according to claim 3, wherein the determination in step d) comprises checking all the nodes of the level *n*+1, and verifying that all the arcs originating from the level *n*+1 have been dealt with.

6. The method according to any of the preceding claims, wherein the method further comprises checking whether there are at least two rules in a same leaf, and this being the case, determining that there is at least one misconfiguration anomaly in the set S of security rules.

7. The method according to claim 6, wherein a console is used to warn an administrator about the different anomalies once a new rule is introduced.

8. The method according to claim 6, wherein the method further comprises warning an administrator managing the electronic device about an anomaly and/or providing a set of rules free of conflicts.

9. The method according to claim 8, wherein providing a set of rules free of conflicts comprises considering priority order of the rules and choosing the rule in a leaf with the highest priority, wherein the corresponding attribute values are obtained by starting from the root and taking the attribute values leading to the current leaf.

10. The method according to claim 8, wherein providing a set of rules free of conflicts comprises determining in each leaf whether there are at least a first rule and a second rule with the same decision and if this is the case then choosing from these rules the rule with the highest priority and if the second rule is preceded in all leaves where it is present by another rule with the same decision then it is detected as completely redundant, and wherein the corresponding attribute values are obtained by starting from the root and taking the attribute values leading to the current leaf.

11. The method according to any of the preceding claims, wherein in case *m* equals to 0, then updating the conflict management tree comprises building a conflict management tree comprising / levels of nodes, / denoting the number of conditional attributes of the rule r, the nodes of different levels being connected to each other by arcs labeled by intervals or subintervals of the conditional attribute values and the nodes being **characterized by** a rule or rules fulfilling the condition set by the interval or subinterval labeling the arc leading to the corresponding node, the conflict management tree having leaves at the end of the arcs originating from the last level of nodes, each leaf being identified by a rule or rules that fulfill the condition set by the interval or subinterval labeling the arc leading to the current leaf and further being identified by (a) corresponding decision(s)

12. A computer program (hardware of software) product comprising instructions for implementing the steps of a method according to any one of claims 1-11 when loaded and run on computer means of the electronic device.

13. An electronic device capable of managing security rule conflicts online in a communication system, wherein the electronic device is arranged to store a set *S* of security rules R_i, and each rule comprising a list of conditional attributes *C* and at least one corresponding decision, the list of conditional attributes further comprising *m* individual conditional attributes C_n identified by corresponding intervals of values for setting a condition, where *n* is a variable identifying different conditional attributes in a rule, the electronic device comprises:

- means for obtaining in case $m \geq 1$ a conflict management tree comprising *m* levels of nodes, the nodes of different levels being connected to each other by arcs labeled by intervals or subintervals of the conditional attributes and the nodes being **characterized by** a rule or rules fulfilling the condition set by the interval or subinterval labeling the arc leading to the corresponding node, the conflict management tree having leaves at the end of the arcs originating from the last level of nodes, each leaf being identified by a rule or rules that fulfill the condition set by the interval or subinterval labeling the arc leading to the current leaf and further being identified by (a) corresponding decision(s);
- receiving unit for receiving a new rule to be added to the set S of security rules; and
- updating unit for updating online the conflict management tree based on the received new rule.

Amended claims in accordance with Rule 137(2) EPC.

1. An online method of managing security rule conflicts in an electronic device arranged to store a set *S* of security rules *R*, each rule comprising a list of conditional attributes *C* and at least one corresponding decision, the list of conditional attributes further comprising *m* individual conditional attributes C_n identified by corresponding intervals of values for setting a condition, where *n* is a variable identifying different conditional attributes in a rule, the method comprises the following steps performed by the electronic device:

- upon determining that $m \geq 1$, obtaining (201) a conflict management tree comprising m levels of nodes, the nodes of different levels being connected to each other by arcs labeled by intervals or subintervals of the conditional attribute values and the nodes being identified by a rule or rules fulfilling the condition set by the interval or subinterval labeling the arc leading to the corresponding node, the conflict management tree having leaves at the end of the arcs originating from the last level of nodes, each leaf being identified by a rule or rules

that fulfill the condition set by the interval or subinterval labeling the arc leading to the current leaf and further being identified by the at least one corresponding decision, wherein at least some of the arcs are labeled by subintervals of intervals of values for setting a condition;
- receiving (203) a new rule r to be added to the set *S* of security rules; and
- updating (229) online the conflict management tree based on the received new rule.

2. The method according to claim 1, wherein the updating comprises:

a) defining (223) a variable *X*;

b) *X* taking the value of $\left(r[C_n]-\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj}\right)\right)\cup$

$$\left(\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj}\right)-r[C_n]\right)\cup\left(r[C_n]\cap\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj}\right)\right),$$ where $\cup$

denotes mathematical operation union, $\cap$ denotes mathematical operation intersection, $current\_node_{arcj}$ denotes the intervals or subintervals of an arc *j* from the current node to the level n-1 node and where

$\left(r[C_n]-\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj}\right)\right)$, $\left(\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj}\right)-r[C_n]\right)$ and

$\left(r[C_n]\cap\left(\bigcup_{j=1}^{number\ of\ arcs} current\_node_{arcj}\right)\right)$ are sets of the union operation corresponding to the intervals or subintervals of the conditional attributes C_n;

c) using (229) the sets of *X* for forming arcs to level *n*-1 nodes so that each set is used to label only one arc, wherein the level *n*-1 nodes are identified by the corresponding rules so that each node is identified only by the rules that fulfill the condition set by the corresponding set labeling the arc leading to the corresponding node.

3. The method according to claim 2, wherein the updating further comprises:

d) determining (231) whether *n*-1 equals to 0; and
e) in case *n*-1 equals to 0, then adding (233) leaves to the level *n*-1 of the current node, the leaves being identified by a rule or rules that fulfill(s) the conditions of the set labeling the arc leading to the corresponding leaf.

4. The method according to claim 3, wherein the updating further comprises:

f) in case *n*-1 does not equal to 0 or after having performed step e), then determining (235) whether all nodes of the current level *n* have been dealt with, and if this is not the case then repeating steps a) to d) until all the nodes of the current level *n* have been dealt with; and
g) decrementing (239) *n* by 1 and then repeating steps a) to f).

5. The method according to claim 3, wherein the determination in step d) comprises checking all the nodes of the level *n*+1, and verifying that all the arcs originating from the level *n*+1 have been dealt with.

6. The method according to any of the preceding claims, wherein the method further comprises checking whether there are at least two rules in a same leaf, and this being the case, determining that there is at least one misconfiguration anomaly in the set *S* of security rules.

7. The method according to claim 6, wherein a console is used to warn an administrator about the different anomalies once a new rule is introduced.

8. The method according to claim 6, wherein the method further comprises warning an administrator managing the electronic device about an anomaly and/or providing a set of rules free of conflicts.

9. The method according to claim 8, wherein providing a set of rules free of conflicts comprises considering priority order of the rules and choosing the rule in a leaf with the highest priority, wherein the corresponding attribute values are obtained by starting from the root and taking the attribute values leading to the current leaf.

10. The method according to claim 8, wherein providing a set of rules free of conflicts comprises determining in each leaf whether there are at least a first rule and a second rule with the same decision and if this is the case then choosing from these rules the rule with the highest priority and if the second rule is preceded in all leaves where it is present by another rule with the same decision then it is detected as completely redundant, and wherein the corresponding attribute values are obtained by starting from the root and taking the attribute values leading to the current leaf.

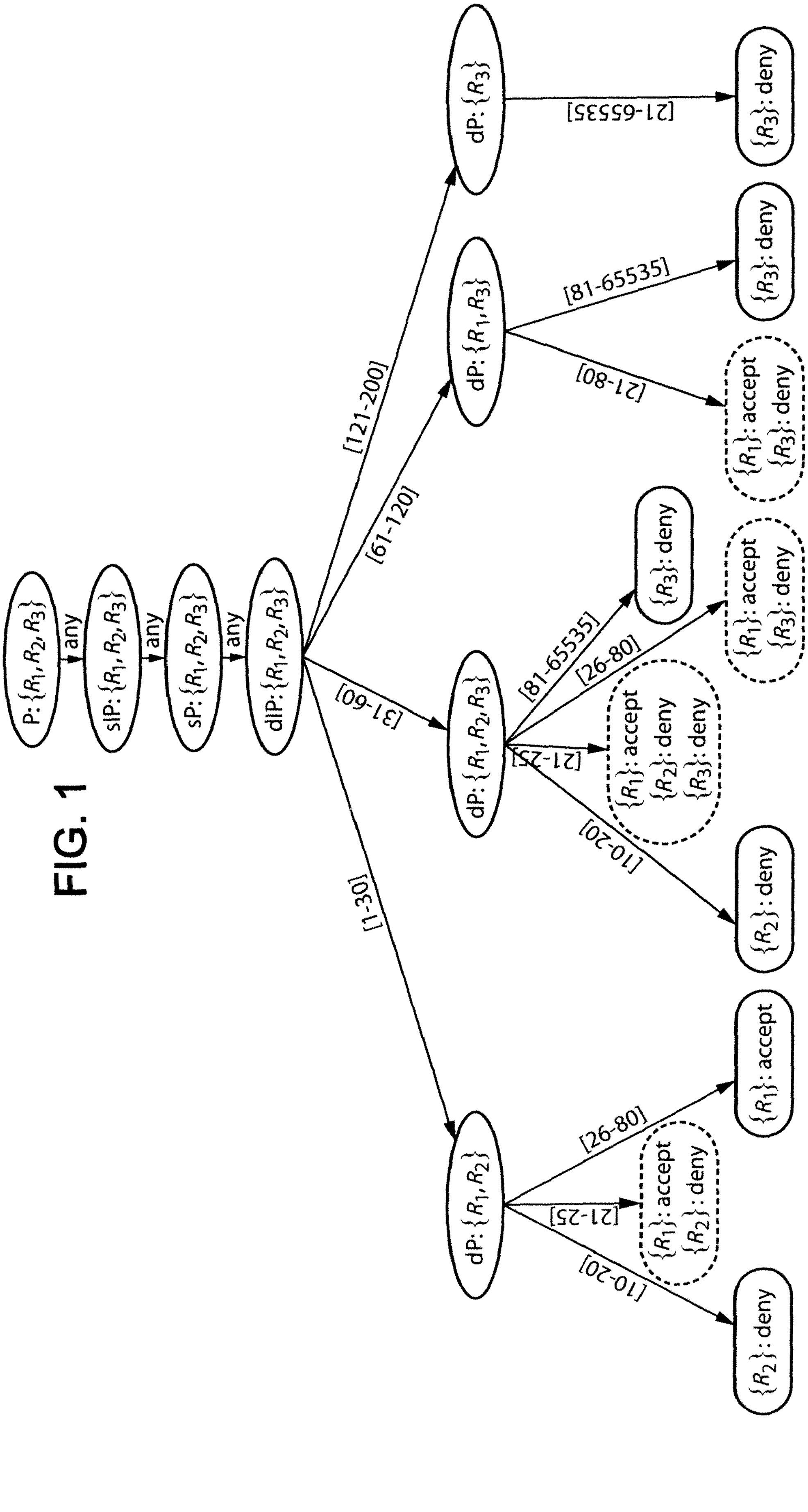
11. The method according to any of the preceding claims, wherein in case m equals to 0, then updating the conflict management tree comprises building a conflict management tree comprising / levels of nodes, / denoting the number of conditional attributes of the rule r, the nodes of different levels being connected to each other by arcs labeled by intervals or subintervals of the conditional attribute values and the nodes being **characterized by** a rule or rules fulfilling the condition set by the interval or subinterval labeling the arc leading to the corresponding node, the conflict management tree having leaves at the end of the arcs originating from the last level of nodes, each leaf being identified by a rule or rules that fulfill the condition set by the interval or subinterval labeling the arc leading to the current leaf and further being identified by (a) corresponding decision(s)

12. A computer program product comprising instructions for implementing the steps of a method according to any one of claims 1-11 when loaded and run on computer means of the electronic device.

13. An electronic device capable of managing security rule conflicts online in a communication system, wherein the electronic device is arranged to store a set S of security rules R_i, and each rule comprising a list of conditional attributes C and at least one corresponding decision, the list of conditional attributes further comprising m individual conditional attributes C_n identified by corresponding intervals of values for setting a condition, where n is a variable identifying different conditional attributes in a rule, the electronic device comprises:

- means for obtaining in case $m \geq 1$ a conflict management tree comprising m levels of nodes, the nodes of different levels being connected to each other by arcs labeled by intervals or subintervals of the conditional attributes and the nodes being identified by a rule or rules fulfilling the condition set by the interval or subinterval labeling the arc leading to the corresponding node, the conflict management tree having leaves at the end of the arcs originating from the last level of nodes, each leaf being identified by a rule or rules that fulfill the condition set by the interval or subinterval labeling the arc leading to the current leaf and further being identified by (a) corresponding decision(s), wherein in the three at least some of the arcs are labeled by subintervals of intervals of values for setting a condition;
- receiving unit for receiving a new rule to be added to the set S of security rules; and
- updating unit for updating online the conflict management tree based on the received new rule.

FIG. 1


P: {R1, R2, R3}
any
sIP: {R1, R2, R3}
any
sP: {R1, R2, R3}
any
dIP: {R1, R2, R3}
[1-30]
[31-60]
[61-120]
[121-200]
dP: {R1, R2}
dP: {R1, R2, R3}
dP: {R1, R3}
dP: {R3}
[10-20]
[21-25]
[26-80]
[81-65535]
[21-80]
[21-65535]
{R2}: deny
{R1}: accept {R2}: deny
{R1}: accept
{R1}: accept {R2}: deny {R3}: deny
{R1}: accept {R3}: deny
{R3}: deny

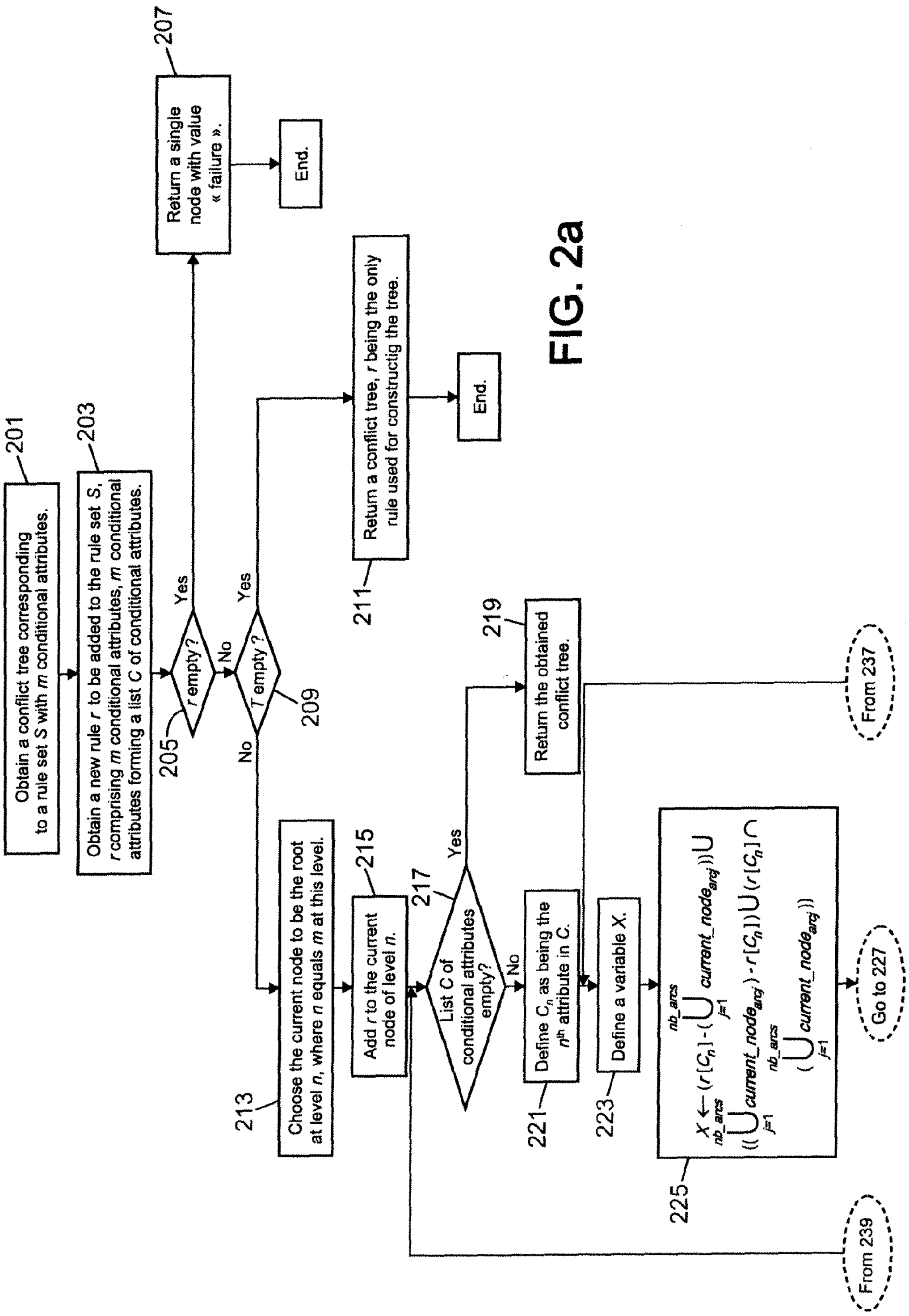

Obtain a conflict tree corresponding to a rule set S with m conditional attributes.
201
Obtain a new rule r to be added to the rule set S, r comprising m conditional attributes, m conditional attributes forming a list C of conditional attributes.
203
r empty?
205
Yes
No
Return a single node with value « failure ».
207
End.
T empty?
209
Yes
No
Return a conflict tree, r being the only rule used for constructig the tree.
211
End.
Choose the current node to be the root at level n, where n equals m at this level.
213
Add r to the current node of level n.
215
List C of conditional attributes empty?
217
Yes
No
Return the obtained conflict tree.
219
Define C_n as being the n^th attribute in C.
221
Define a variable X.
223
225
From 237
Go to 227
From 239

FIG. 2a

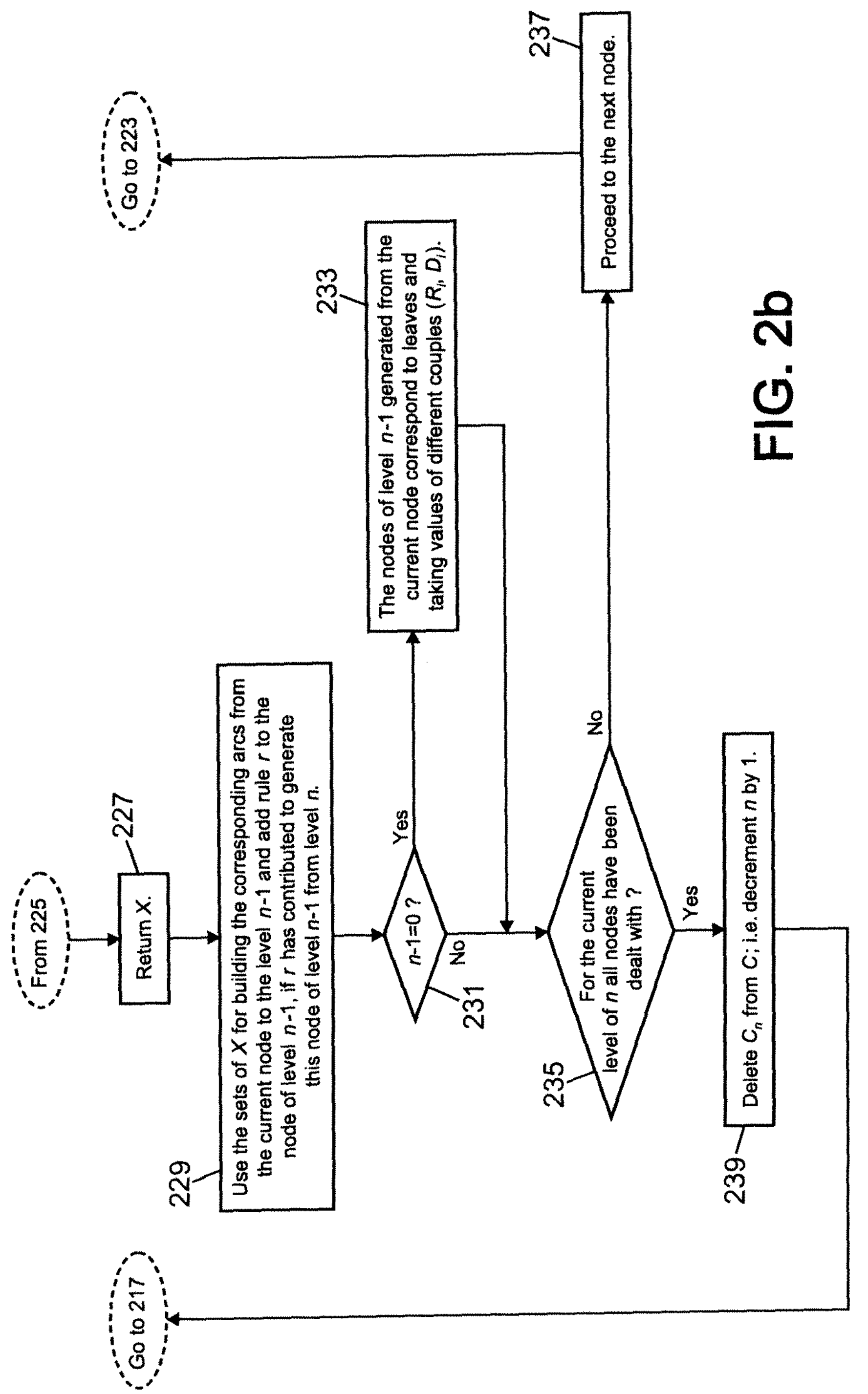

From 225
Return X.
227
Use the sets of X for building the corresponding arcs from the current node to the level n-1 and add rule r to the node of level n-1, if r has contributed to generate this node of level n-1 from level n.
229
n-1=0 ?
231
Yes
No
The nodes of level n-1 generated from the current node correspond to leaves and taking values of different couples (R_i, D_i).
233
For the current level of n all nodes have been dealt with ?
235
No
Yes
Proceed to the next node.
237
Go to 223
Delete C_n from C; i.e. decrement n by 1.
239
Go to 217

FIG. 2b

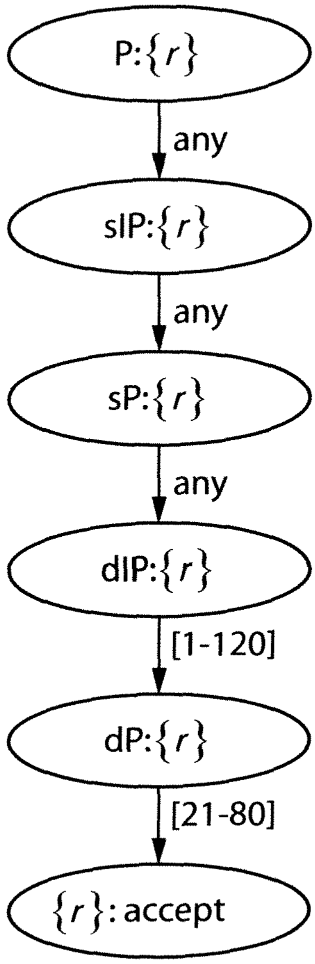
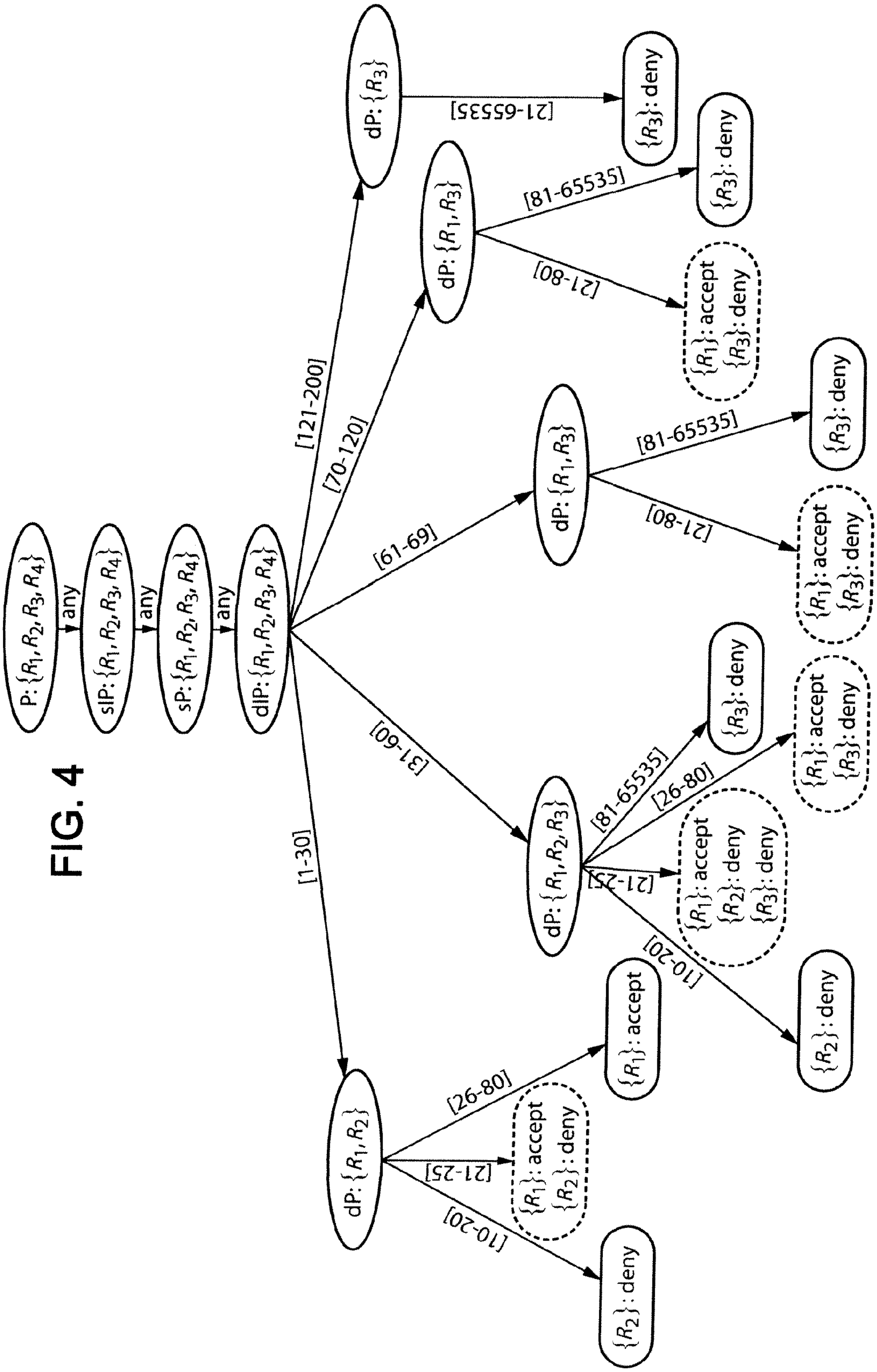


FIG. 3

FIG. 4


P: {R1, R2, R3, R4}
any
sIP: {R1, R2, R3, R4}
any
sP: {R1, R2, R3, R4}
any
dIP: {R1, R2, R3, R4}
[1-30]
[31-60]
[61-69]
[70-120]
[121-200]
dP: {R1, R2}
[10-20]
[21-25]
[26-80]
{R2}: deny
{R1}: accept
{R2}: deny
{R1}: accept
dP: {R1, R2, R3}
[10-20]
[21-25]
[26-80]
[81-65535]
{R2}: deny
{R1}: accept
{R2}: deny
{R3}: deny
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R1, R3}
[21-80]
[81-65535]
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R1, R3}
[21-80]
[81-65535]
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R3}
[21-65535]
{R3}: deny

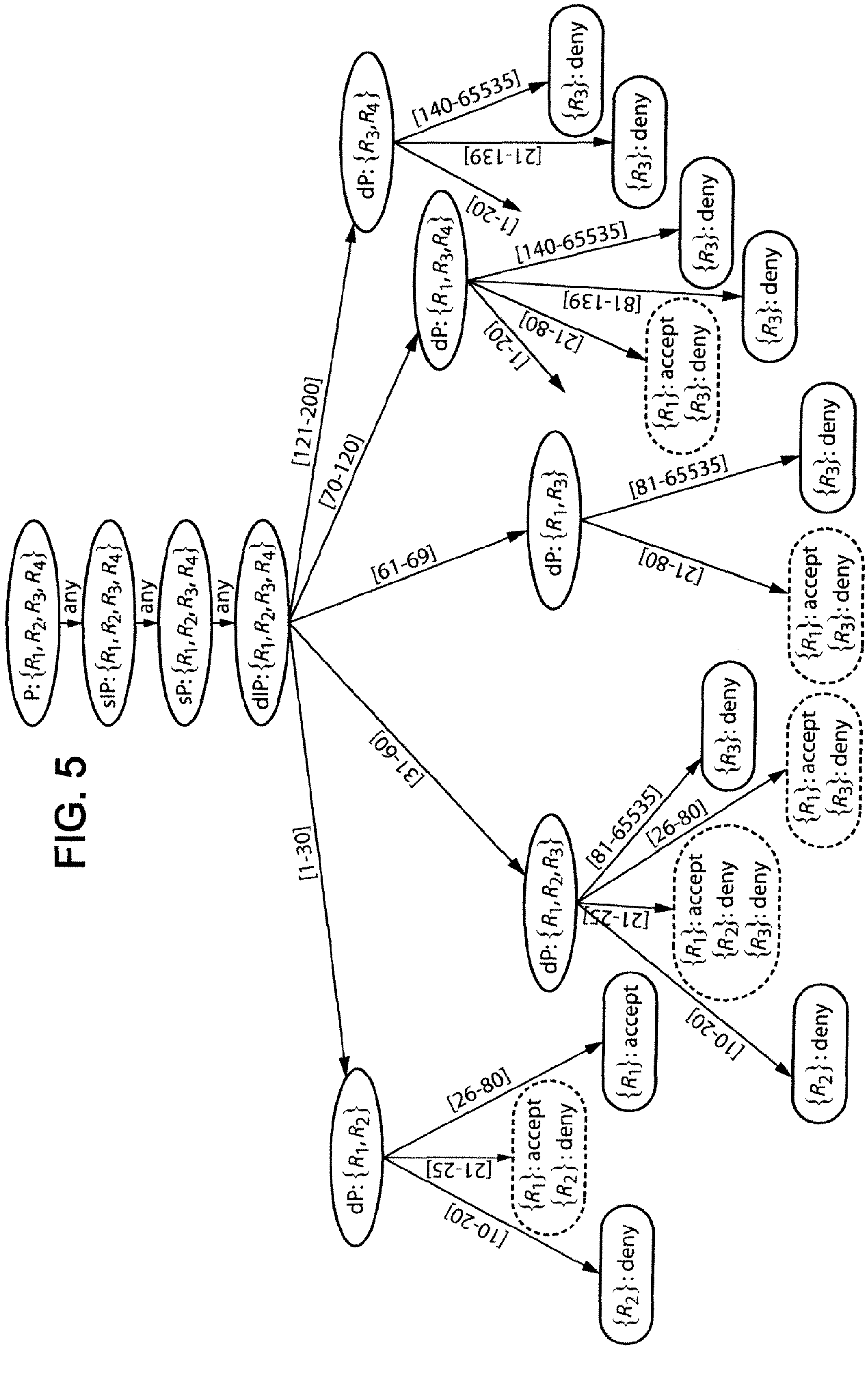

P: {R1, R2, R3, R4}
any
sIP: {R1, R2, R3, R4}
any
sP: {R1, R2, R3, R4}
any
dIP: {R1, R2, R3, R4}
[1-30]
[31-60]
[61-69]
[70-120]
[121-200]
dP: {R1, R2}
[10-20]
[21-25]
[26-80]
{R2}: deny
{R1}: accept
{R2}: deny
{R1}: accept
dP: {R1, R2, R3}
[10-20]
[21-25]
[26-80]
[81-65535]
{R2}: deny
{R1}: accept
{R2}: deny
{R3}: deny
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R1, R3}
[21-80]
[81-65535]
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R1, R3, R4}
[1-20]
[21-80]
[81-139]
[140-65535]
{R1}: accept
{R3}: deny
{R3}: deny
{R3}: deny
dP: {R3, R4}
[1-20]
[21-139]
[140-65535]
{R3}: deny
{R3}: deny

FIG. 5

FIG. 6

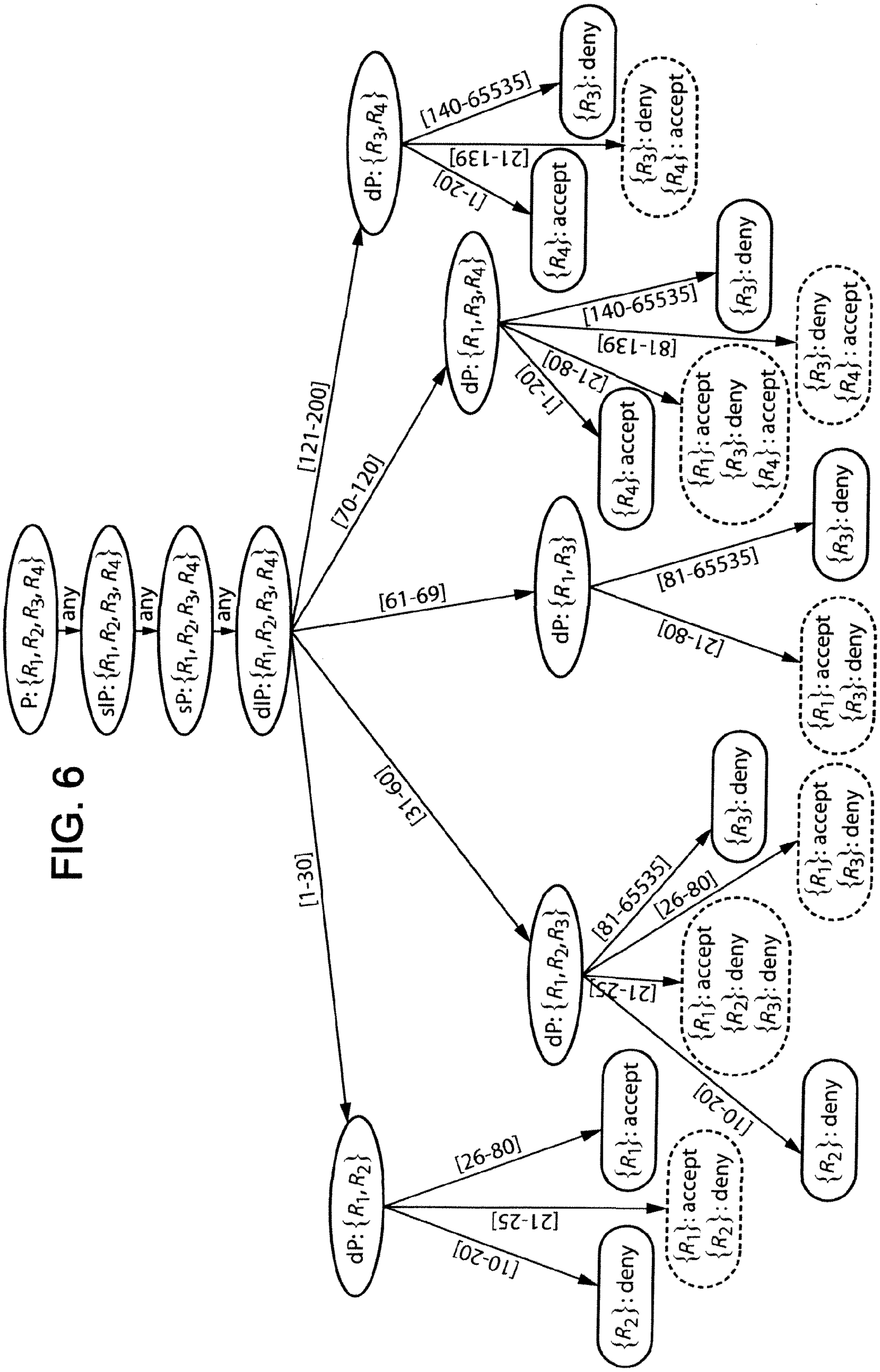
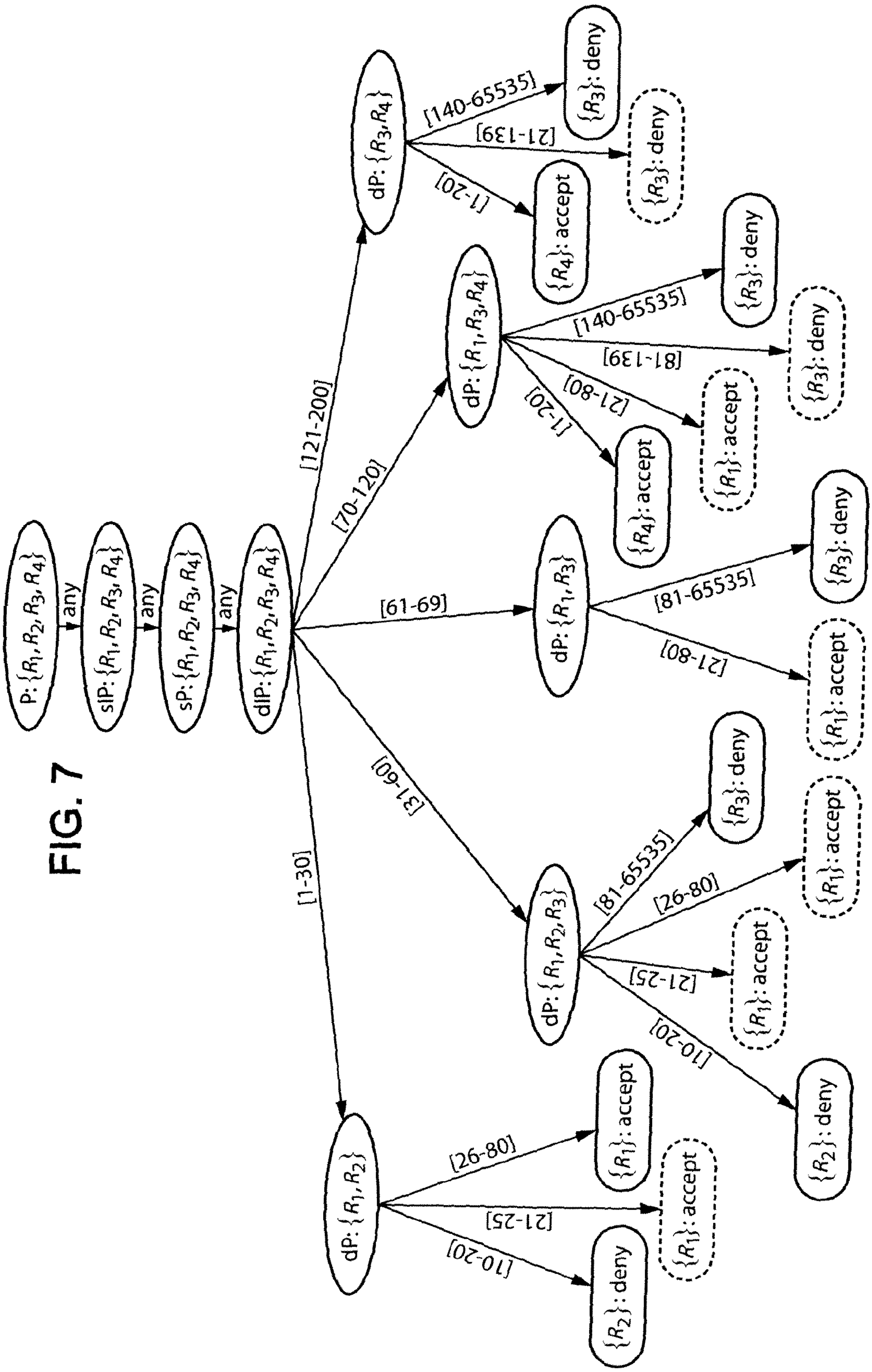

P: {R1, R2, R3, R4}
any
sIP: {R1, R2, R3, R4}
any
sP: {R1, R2, R3, R4}
any
dIP: {R1, R2, R3, R4}
[1-30]
[31-60]
[61-69]
[70-120]
[121-200]
dP: {R1, R2}
[10-20]
[21-25]
[26-80]
{R2}: deny
{R1}: accept
{R2}: deny
{R1}: accept
dP: {R1, R2, R3}
[10-20]
[21-25]
[26-80]
[81-65535]
{R2}: deny
{R1}: accept
{R2}: deny
{R3}: deny
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R1, R3}
[21-80]
[81-65535]
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R1, R3, R4}
[1-20]
[21-80]
[81-139]
[140-65535]
{R4}: accept
{R1}: accept
{R3}: deny
{R4}: accept
{R3}: deny
{R4}: accept
{R3}: deny
dP: {R3, R4}
[1-20]
[21-139]
[140-65535]
{R4}: accept
{R3}: deny
{R4}: accept
{R3}: deny

FIG. 7


P: {R1, R2, R3, R4}
any
sIP: {R1, R2, R3, R4}
any
sP: {R1, R2, R3, R4}
any
dIP: {R1, R2, R3, R4}
[1-30]
[31-60]
[61-69]
[70-120]
[121-200]
dP: {R1, R2}
[10-20]
[21-25]
[26-80]
{R2}: deny
{R1}: accept
{R1}: accept
dP: {R1, R2, R3}
[10-20]
[21-25]
[26-80]
[81-65535]
{R2}: deny
{R1}: accept
{R1}: accept
{R3}: deny
dP: {R1, R3}
[21-80]
[81-65535]
{R1}: accept
{R3}: deny
dP: {R1, R3, R4}
[1-20]
[21-80]
[81-139]
[140-65535]
{R4}: accept
{R1}: accept
{R3}: deny
{R3}: deny
dP: {R3, R4}
[1-20]
[21-139]
[140-65535]
{R4}: accept
{R3}: deny
{R3}: deny

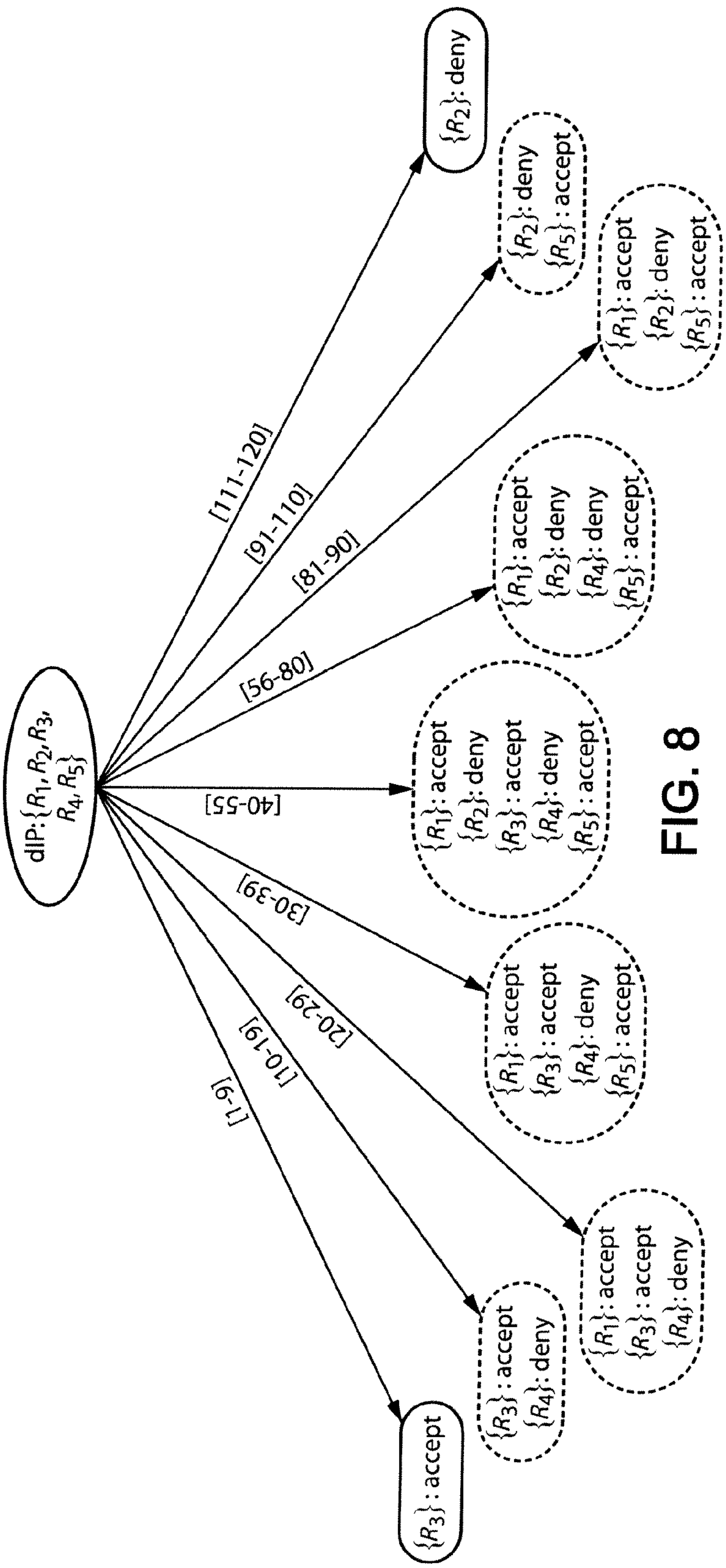

dIP: {R1, R2, R3, R4, R5}
[111-120]
{R2}: deny
[91-110]
{R2}: deny
{R5}: accept
[81-90]
{R1}: accept
{R2}: deny
{R5}: accept
[56-80]
{R1}: accept
{R2}: deny
{R4}: deny
{R5}: accept
[40-55]
{R1}: accept
{R2}: deny
{R3}: accept
{R4}: deny
{R5}: accept
[30-39]
{R1}: accept
{R3}: accept
{R4}: deny
{R5}: accept
[20-29]
{R1}: accept
{R3}: accept
{R4}: deny
[10-19]
{R3}: accept
{R4}: deny
[1-9]
{R3}: accept

FIG. 8

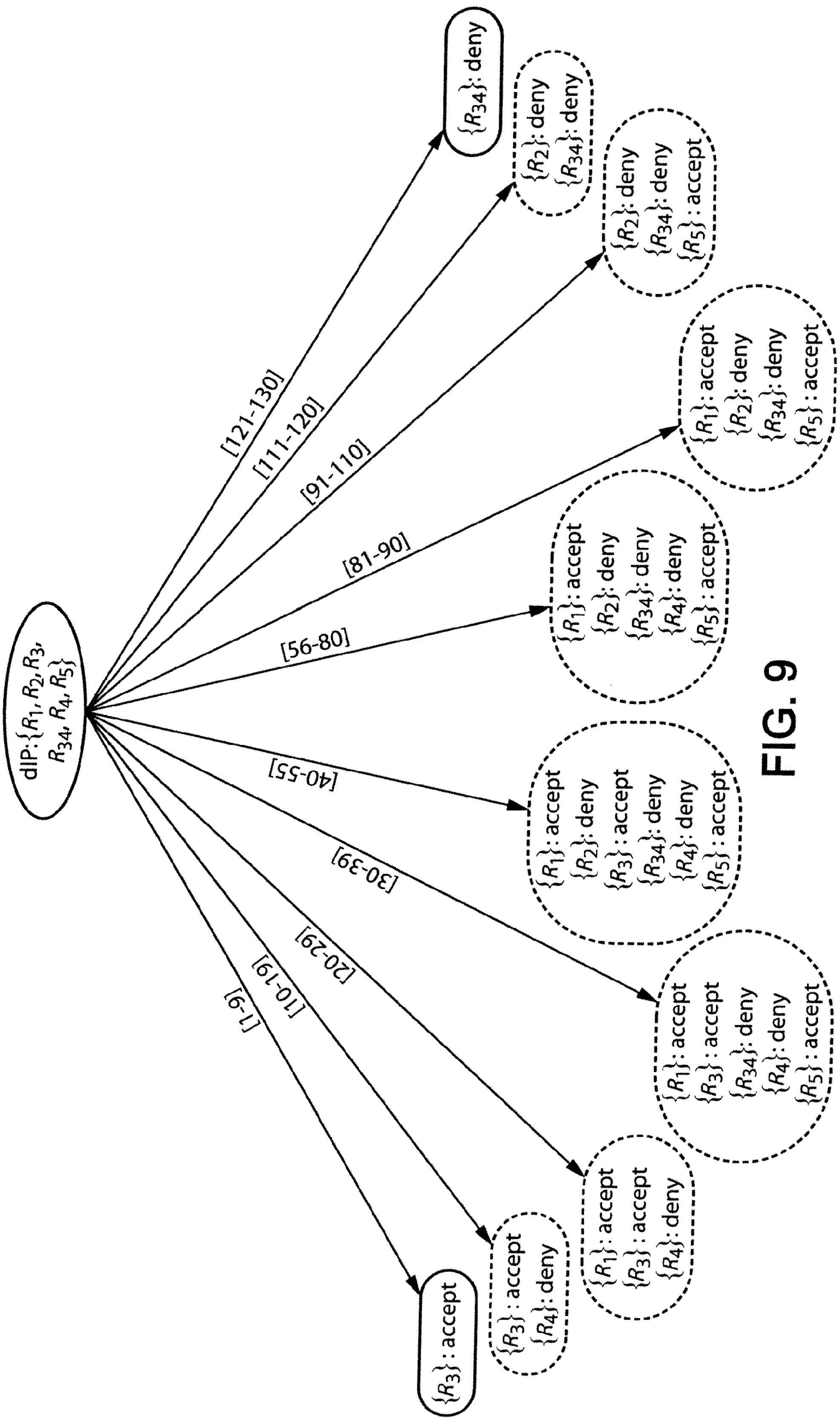
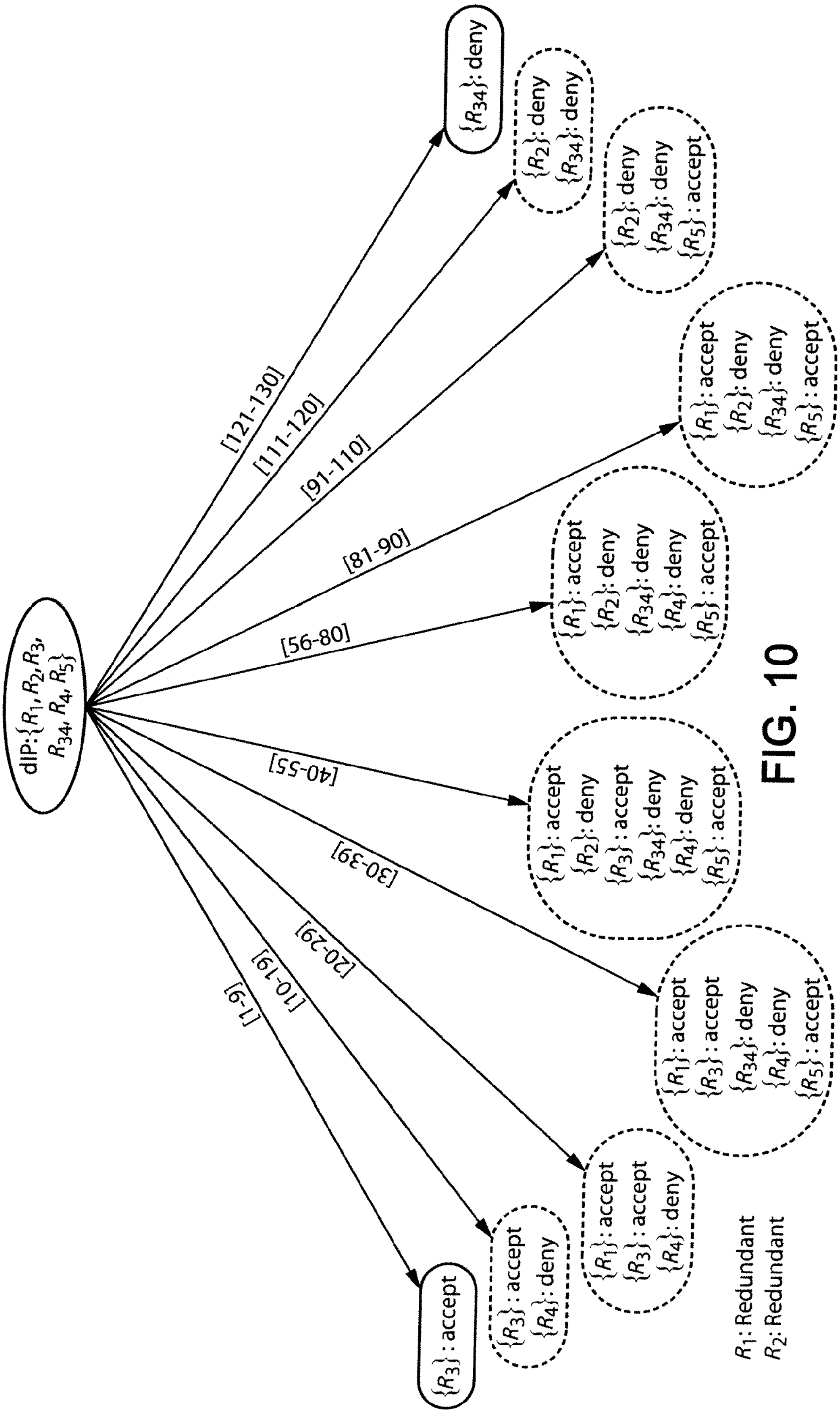

dIP: {R1, R2, R3, R34, R4, R5}
[1-9]
[10-19]
[20-29]
[30-39]
[40-55]
[56-80]
[81-90]
[91-110]
[111-120]
[121-130]
{R3} : accept
{R3} : accept
{R4}: deny
{R1}: accept
{R3} : accept
{R4}: deny
{R1}: accept
{R3} : accept
{R34}: deny
{R4}: deny
{R5}: accept
{R1}: accept
{R2}: deny
{R3}: accept
{R34}: deny
{R4}: deny
{R5}: accept
{R1}: accept
{R2}: deny
{R34}: deny
{R4}: deny
{R5}: accept
{R1}: accept
{R2}: deny
{R34}: deny
{R5}: accept
{R2}: deny
{R34}: deny
{R5}: accept
{R2}: deny
{R34}: deny
{R34}: deny

FIG. 9



dIP: $\{R_1, R_2, R_3, R_{34}, R_4, R_5\}$
[1-9]
[10-19]
[20-29]
[30-39]
[40-55]
[56-80]
[81-90]
[91-110]
[111-120]
[121-130]
$\{R_3\}$: accept
$\{R_3\}$: accept
$\{R_4\}$: deny
$\{R_1\}$: accept
$\{R_3\}$: accept
$\{R_4\}$: deny
$\{R_1\}$: accept
$\{R_3\}$: accept
$\{R_{34}\}$: deny
$\{R_4\}$: deny
$\{R_5\}$: accept
$\{R_1\}$: accept
$\{R_2\}$: deny
$\{R_3\}$: accept
$\{R_{34}\}$: deny
$\{R_4\}$: deny
$\{R_5\}$: accept
$\{R_1\}$: accept
$\{R_2\}$: deny
$\{R_{34}\}$: deny
$\{R_4\}$: deny
$\{R_5\}$: accept
$\{R_1\}$: accept
$\{R_2\}$: deny
$\{R_{34}\}$: deny
$\{R_5\}$: accept
$\{R_2\}$: deny
$\{R_{34}\}$: deny
$\{R_5\}$: accept
$\{R_2\}$: deny
$\{R_{34}\}$: deny
$\{R_{34}\}$: deny
R_1: Redundant
R_2: Redundant

FIG. 10

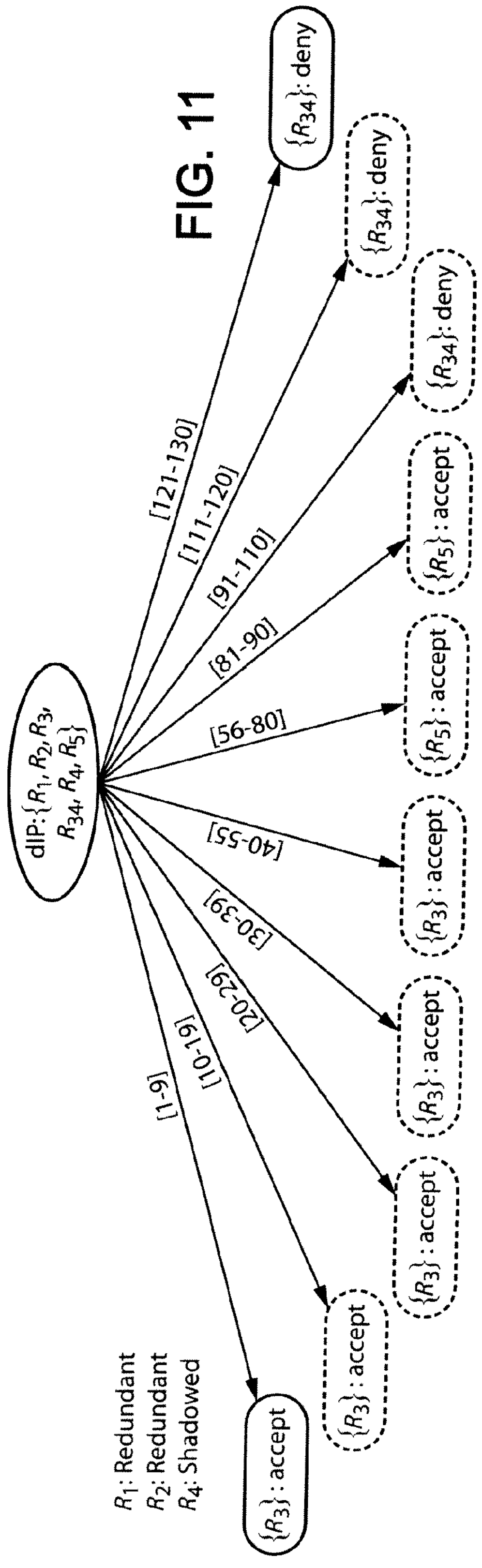

dIP:{R1, R2, R3, R34, R4, R5}
[1-9]
[10-19]
[20-29]
[30-39]
[40-55]
[56-80]
[81-90]
[91-110]
[111-120]
[121-130]
{R3}: accept
{R3}: accept
{R3}: accept
{R3}: accept
{R3}: accept
{R5}: accept
{R5}: accept
{R34}: deny
{R34}: deny
{R34}: deny
R1: Redundant
R2: Redundant
R4: Shadowed

FIG. 11

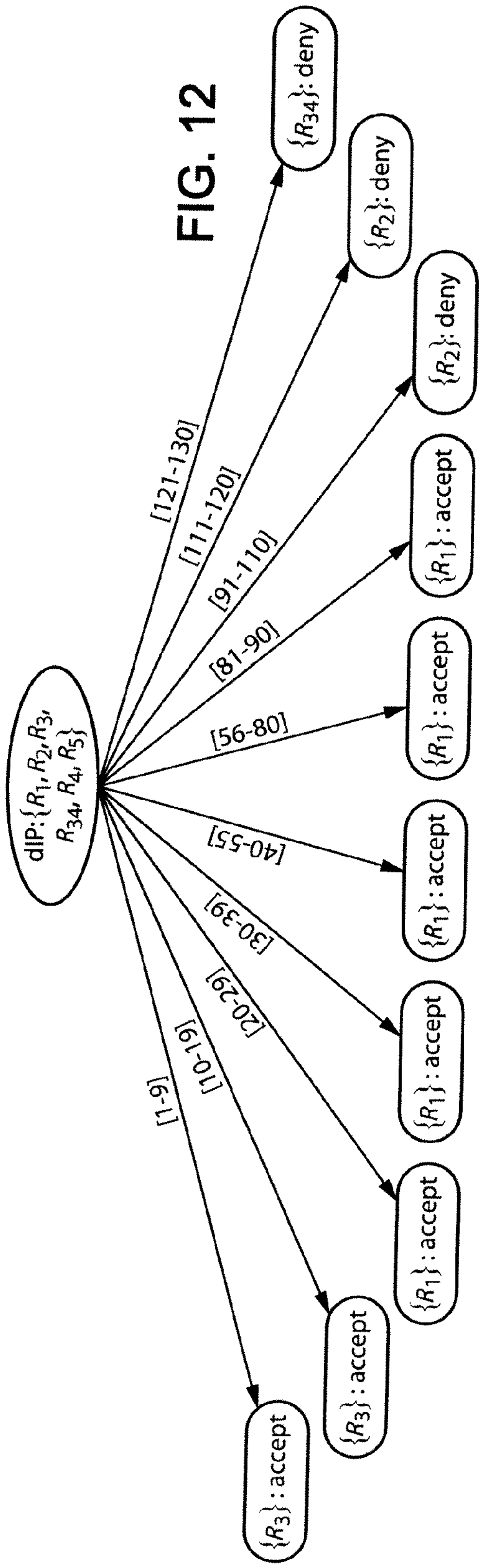

dIP:{R1, R2, R3, R34, R4, R5}
[1-9]
[10-19]
[20-29]
[30-39]
[40-55]
[56-80]
[81-90]
[91-110]
[111-120]
[121-130]
{R3}: accept
{R3}: accept
{R1}: accept
{R1}: accept
{R1}: accept
{R1}: accept
{R1}: accept
{R2}: deny
{R2}: deny
{R34}: deny

FIG. 12

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 07 11 4046

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	EHAB S. AL-SHAER, HAZEM H. HAMED: "FIREWALL POLICY ADVISOR FOR ANOMALY DISCOVERY AND RULE EDITING"[Online] 4 April 2003 (2003-04-04), XP002463654 Multimedia Networking Research Laboratory School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, USA Retrieved from the Internet: URL:http://www.mnlab.cs.depaul.edu/~ehab/papers/im03-cr.pdf> [retrieved on 2008-01-07]	1,3-13	INV. H04L29/06
Y	* pages 3-4, paragraph 3.1; figure 2 * * pages 5-6, paragraph 3.2 * * pages 10-11, paragraph 5.1; figures 6,7 * * page 12, paragraph 7 * -----	2	
X	AL-SHAER E ET AL: "Conflict Classification and Analysis of Distributed Firewall Policies" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE SERVICE CENTER, PISCATAWAY, US, vol. 23, no. 10, October 2005 (2005-10), pages 2069-2084, XP011140074 ISSN: 0733-8716	1,3-13	TECHNICAL FIELDS SEARCHED (IPC) H04L
A	* page 2070, right-hand column, paragraph A; figure 1 * * page 2071, right-hand column, paragraph B; figure 2 * * page 2079, left-hand column, paragraph 2 - right-hand column, paragraph 2 * ----- -/--	2	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	8 January 2008	Pajatakis, Emmanouil

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

4

EPO FORM 1503 03.82 (P04C01)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 07 11 4046

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X,D	AL-SHAER E S ET AL: "Discovery of policy anomalies in distributed firewalls" INFOCOM 2004. TWENTY-THIRD ANNUALJOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES HONG KONG, PR CHINA 7-11 MARCH 2004, PISCATAWAY, NJ, USA,IEEE, vol. 4, 7 March 2004 (2004-03-07), pages 2605-2616, XP010740626 ISBN: 0-7803-8355-9	1,3-13	
A	* page 2606, right-hand column, paragraph A * * page 2607, right-hand column, paragraph B; figure 2 * * page 2613, paragraph A * -----	2	
Y	US 2006/294577 A1 (GOUDA MOHAMED G [US] ET AL) 28 December 2006 (2006-12-28) * paragraphs [0028], [0029], [0031] - [0039], [0051], [0055]; figures 4,5 * -----	2	
			TECHNICAL FIELDS SEARCHED (IPC)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	8 January 2008	Pajatakis, Emmanouil

4

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 07 11 4046

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

08-01-2008

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006294577 A1	28-12-2006	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Non-patent literature cited in the description

- **EHAB S. AL-SHAER ; HAZEM H. HAMED.** Discovery of Policy Anomalies in Distributed Firewalls. *IEEE INFOCOM,* 2004 **[0007]**
- **F. CUPPENS ; N. CUPPENS-BOULAHIA ; J. GARCÍA-ALFARO.** Detection and Removal of Firewall Misconfiguration. *IASTED International Conference on Communication, Network and Information Security,* 2005 **[0010]**