Recognizing Immediacy in an N-Tree Hierarchy and Its Application to Protection Groups

RAVINDERPAL S. SANDHU, MEMBER, IEEE

Abstract—**The benefits of providing access control with groups of users as the unit of granularity are well known. These benefits are enhanced if the groups are organized in a hierarchy (partial order) by the subgroup relation $\leq$, where $g \leq h$ signifies that every member of group g is thereby also a member of group h. It is often useful to distinguish the case when g is an *immediate subgroup* of h, that is when $g < h$ and there is no group k such that $g < k < h$. The class of partial orders called n-trees was recently defined by using rooted trees and inverted rooted trees as basic partial orders and combining these recursively by refinement [12]. It has been shown that n-trees arise naturally in many practical situations and they have a simple representation. Any n-tree hierarchy can be expressed as the intersection of two linear orderings. So it is possible to assign a pair of integers $l[x]$ and $r[x]$ to each group x such that $g \leq h$ if and only if $l[g] \leq l[h]$ and $r[g] \leq r[h]$. In this paper we show how to extend this representation of n-trees by assigning four additional integers to each group so that it is also easily determined whether or not g is an immediate subgroup of h.**

Index Terms—**Access control, access control lists, authorization, protection, protection groups, security.**

I. Introduction

THE ability to share files and other resources among the users of a system has obvious benefits. It is convenient for both users and system administrators to have a facility to specify access based on groups of users as a unit. Since membership in a group is presumably determined by the need to share resources and information, the group provides a suitable unit for an individual user's access decisions. A user can make a file available to an entire group without having to explicitly provide access to every member. Similarly, a user can revoke a file's availability from an entire group without explicitly revoking each member's access to the file. Also, new users can be made members of appropriate groups, thereby obtaining access to a number of files and resources. Some systems such as the popular Unix [10] allow for access control only in terms of groups. Even the more sophisticated systems such as Multics [11], which have provision for specifying access at the level of individual users, recognize the advantages of protection groups and provide facilities for specifying access in terms of groups.

In practice, it is often desirable that groups bear some relationship to each other. For instance, consider a project divided into several independent tasks assigned to different terms. We can define a group for each task team so its members have common access to resources relevant to the task. Since some resources may pertain to the entire project, we can define a project group such that members of the individual task groups are thereby also members of the project group. The project-wide resources are then made explicitly available to the project group alone. This is certainly more convenient than having to explicitly make such resources available to every task group, even if it were possible to do so. It is also more convenient than explicitly making every member of a task group a member of the project group. By allowing membership in a group to automatically imply membership in some other groups, we can reduce the number of explicit access decisions that need to be made by the users, as well as reduce the number of groups to which a user must explicitly belong.

Let G be a set of groups and let $g \leq h$ signify that group g is a *subgroup* of group h, in the sense that every member of g is thereby also a member of h. If g is a proper subgroup of h we write $g < h$, that is $g \leq h$ and $g \neq h$. We say a user is a *direct* member of g if the user is explicitly designated as a member of g and thereby is an *indirect* member of every h such that $g < h$. The intention is that a user will be a direct member of a small number of unrelated groups, perhaps just one, but will thereby obtain indirect membership in a larger number of groups.

We require that the subgroup relation is a partial ordering of G, that is $\leq$ is a reflexive, transitive, and asymmetric binary relation on G. The reflexive property is obviously required since every member of g is already a member of g. Transitivity is certainly an intuitive and reasonable assumption and perhaps even inevitable. After all, if $g < h$ and $h \leq k$ then every direct member of g is an indirect member of h and so should also be an indirect member of k. The asymmetric requirement merely eliminates redundancy by excluding groups which would otherwise be equivalent.

Our objective is that once the access control mechanism knows that a user is a direct member of group g, it should be easily determined whether the user is therefore an indirect member of some other group h. To do so we must represent the subgroup partial order so it is easy to determine whether one group is a subgroup of another. A class of partial orders called n-trees has been recently proposed by this author [12] and it has been shown that n-trees have

Manuscript received December 3, 1986; revised August 18, 1989. Recommended by V. Gligor.

The author is with the Department of Information Systems and Systems Engineering, George Mason University, Fairfax, VA 22030.

IEEE Log Number 8931482.

a representation which meets this objective. The practical application of n-trees has also been demonstrated by examples which show that these hierarchies arise naturally in many situations [12].

In this paper we show that the representation for n-trees can be extended to make some important finer distinctions regarding the relationship between two groups. We say that g is an *immediate subgroup* of h if $g < h$ and there is no group k such that $g < k < h$. In such cases we call g an *immediate predecessor* of h and h an *immediate successor* of g. We propose a technique for representing n-trees which allows this distinction to be easily made.

Distinguishing immediacy is useful because in many practical situations there is considerably stronger affinity between a group and its immediate successors or predecessors as compared to the nonimmediate ones. For instance, consider a corporate hierarchy with various divisions, departments and projects at successive levels of the hierarchy. It is clearly of practical value to identify a particular division and its immediate successors, viz. the department groups within that division, as a unit for access control purposes while excluding the project groups, which are nonimmediate successors of the division group. This enables the management of the division and its departments to share information which is kept confidential from their project staff. In a complementary manner a group and its immediate predecessors can be usefully treated as a unit for acess control. In practical situations the immediate predecessors of a group often have direct responsibility for supervising that group's activities. Because of this close working relationship it is therefore appropriate to have a means by which one can share information with one's immediate predecessors while excluding nonimmediate ones. So members of a project which reports to multiple departments should be able to share files with these department groups, which are immediate predecessors, while excluding the nonimmediate division groups. We will shortly consider more concrete examples of access-control policies based on immediacy.

The paper is organized as follows. In Section II we review the concept of n-trees [12]. An n-tree is defined recursively by using a forest of rooted trees or inverted rooted trees as basic partial orders and combining these by an operation called refinement. We discuss examples to show the practical importance of n-trees for protection groups. In Section III we further pursue the reasoning outlined above to establish that recognizing immediacy in a hierarchy is a useful feature for access-control policies. Section IV goes on to review the property that an n-tree can be represented as the intersection of two linear orderings [12]. So it is possible to assign a pair of integers $l[x]$ and $r[x]$ to each group x such that $g < h$ if and only if $l[g] < l[h]$ and $r[g] < r[h]$. The main result of this paper, presented in Section V, is to extend the representation of n-trees by assigning four additional integers to each group so it is also easily determined whether or not g is an immediate subgroup of h. Section VI concludes the paper.

II. N-Tree Partial Orders

Partial orders are conventionally depicted by Hasse diagrams as shown in Fig. 1 for instance. The partial order represented by a Hasse diagram is obtained by directing the edges downwards, for example from a to b in Fig. 1(a) indicating $a < b$, and taking the reflexive transitive closure of the resulting directed graph.

The simplest and most familiar partial orders are the rooted tree and its dual, the inverted rooted tree. These represent important relationships between groups which have practical application. Consider a project divided into three independent tasks with each task assigned to a team. We can define groups t_1, t_2, and t_3 for the tasks and a group s for the project supervisors related as in Fig. 2(a) so the supervisors are members of each task team but not vice versa. This tree allows the information and resources such as working documents for each task group to be kept separate and inaccessible from other task groups while a supervisor can access all of these. Alternatively we can define a single group p related to the task groups as shown in Fig. 2(b). With this inverted tree the task teams can share information and resources of common interest, such as the final design produced by a task team, while keeping working documents within each task group. Finally, the tree and inverted tree are not only useful by themselves but can occur together as in Fig. 2(c).

The partial order of Fig. 2(c) is an example of the class of partial orders called *n-trees*. The ntree embodies three important aspects of a protection policy.

1) Separation: The three task groups t_1, t_2, and t_3 are pairwise incomparable with respect to the subgroup ordering.

2) Sharing: The three separate task groups are all subgroups of a common group p which allows sharing of information and resources.

3) Oversight: The three separate task groups all have s as a common subgroup to facilitate oversight and coordination.

Independent groups which are pairwise incomparable provide support only for separation. A tree can support separation and oversight while an inverted tree supports separation and sharing. The n-tree partial order supports all three aspects.

N-trees are constructed by using rooted trees and inverted rooted trees as basic partial orders and combining these recursively by the operation of *refinement* defined as follows. Let P and Q be partial orders on disjoint sets G and H, respectively. Consider some $u \in G$. The refinement of u in P into Q is the partial order P' on the set $(G - \{u\}) \cup H$ formed by the union of the following sets of ordered pairs.

1) $\{(x, x') \mid (x, x') \in P \text{ for all } x, x' \in G - \{u\}\}$
2) $\{(x, y) \mid (x, u) \in P \text{ for all } x \in G - \{u\}, y \in H\}$
3) $\{(y, x) \mid (u, x) \in P \text{ for all } x \in G - \{u\}, y \in H\}$
4) $\{(y, y') \mid (y, y') \in Q \text{ for all } y, y' \in H\}$

Fig. 3(c) shows the result of refining b in the partial order of Fig. 3(a) into the partial order of Fig. 3(b). Informally,

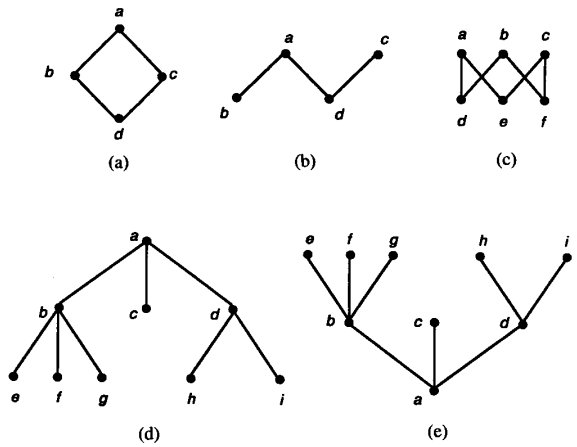


Fig. 1. Hasse diagrams.

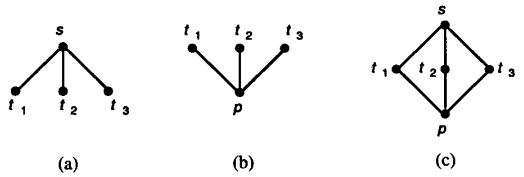


Fig. 2. Subgroup partial orders for a project.

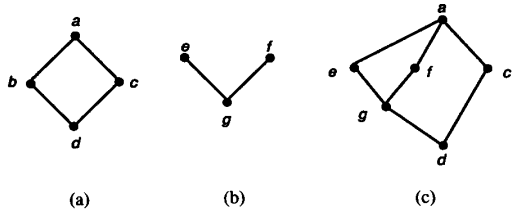


Fig. 3. An example of refinement.

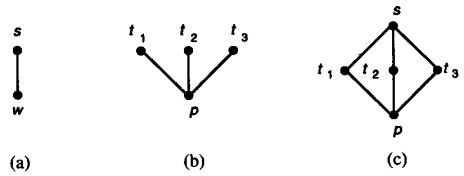


Fig. 4. Construction of an n-tree by refinement.

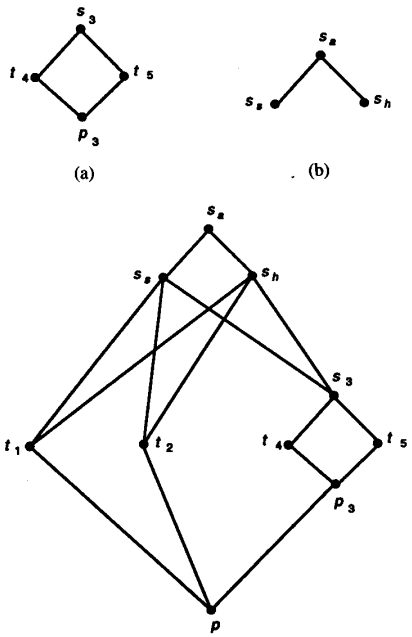


Fig. 5. Further refinement of the project n-tree.

the refinement of u in P into Q is the partial order whose Hasse diagram is obtained by substituting Q's Hasse diagram in place of u in P's Hasse diagram. We think of refinement as exploding an existing group into a partially ordered set of new groups while maintaining the same relationship between the new groups and other previously existing groups which the exploded group had. Refinement is a natural method for incrementally developing more detail in a top-down manner.

The refinement operation is used to define the class of partial orders called n-trees as follows.

1) A partial order whose Hasse diagram is a forest of mutually disjoint rooted trees and inverted rooted trees is an n-tree.

2) A partial order obtained by refining a node in an n-tree into another n-tree is an n-tree.

3) Nothing else is an n-tree.

The n in the name n-tree is intended as a mnemonic both for inverted and for nested in the sense of refinement. Fig. 4 shows one method of constructing the n-tree of Fig. 2(c) by refinement. We begin with two groups, s for supervisors, and w for workers as in Fig. 4(a). Then w is refined into the three task groups and project group of Fig. 4(b) to obtain the n-tree of Fig. 4(c).

N-trees constitute a rich class of subgroup relations of practical importance. The advantage of adopting the tree and inverted tree as the basis for generating complex partial orders is that incremental policy decisions are easy to understand. Repeated application of the refinement operation provides a natural method for constructing a complex n-tree in a top-down manner. For instance if it turns out that task t_3 of Fig. 4(c) is complex enough to justify treating it as a subproject the group t_3 can be refined into the n-tree of Fig. 5(a) with two distinct task groups t_4 and t_5 and supervisory and project groups s_3 and p_3. At the same time the supervisory group s of Fig. 4(c) may be refined into the tree of Fig. 5(b) with separate supervisory groups s_s and s_h for software and hardware respectively and an overall supervisory group s_a at the root. The n-tree obtained after these two refinements is shown in Fig. 5(c). It is evident that it would be quite difficult to arrive at this result directly while the sequence of refinements indicated in Figs. 4 and 5 do so by straightforward incremental steps.

As a final note we wish to point out that there are partial orders of practical importance which are not n-trees. Most notably the partial order obtained by ordering the subsets of a set S by set inclusion fails to be an n-tree for $|S| > 2$. Other examples are discussed in [12]. We will see in Section V that the partial order of Fig. 1(b) fails to be an n-tree.

III. Immediacy and Access Control

In Section I we outlined the basic reason why immediacy in a hierarchy is an important issue for access control policies. That is, there is considerably stronger affinity between a group and its immediate successors or predecessors as compared to the nonimmediate ones. To make this point more concrete let us consider some policies based on immediacy in context of the n-tree of Fig. 5(c). For simplicity assume each user is a direct member of exactly one group. Of course, he thereby obtains indirect membership in all groups which are successors of his direct group.

Now the supervisory groups at the top of the n-tree are s_a, s_s, and s_h. It is very reasonable that the direct members of these three groups would like to share files while excluding the rest of the hierarchy. The most natural way to do so is to associate access control information with such files, perhaps in an access control list, saying that the file is accessible by the direct members of s_a and its immediate successors. One might be able to achieve the same result by explicitly enumerating all three groups, s_a, s_s, and s_h, in the access control list with the stipulation that only the direct members of these three groups have access to the file. This latter approach has several disadvantages as compared to the former. Principally if the hierarchy is later changed to include a third child of s_a as another supervisory group, by say refining s_s into two incomparable groups, we would need to update the access control list on all such files to reflect this change. In the former approach additional children of s_a would be automatically included in the access set without any modification of the access control lists. The latter approach moreover obscures the intent of the access control decision. The same observations apply to s_a and its immediate successors, t_1, t_2, and s_3, and so on.

In a dual manner, looking upwards in the hierarchy, there is practical value in the ability to share files with the immediate predecessors of a group. For example the direct members of task t_1 presumably report to the direct members of s_s and s_h, and should therefore be able to share files among these three groups while excluding everyone else. Similarly direct members of t_4 should be able to share files with their supervisors, i.e., the direct members of s_3, confidentially from all other groups.

We now consider a more subtle issue, concerning limits[1] on the discretionary ability of a user to make access control decisions regarding his own files. The intuitive policy we wish to implement is that information should be "passed upward" in the hierarchy in a very controlled manner. Note that the groups at the top of the hierarchy are the most powerful regarding the ability of their members to access files. So the direct members of s_a potentially have access to a large amount of information. The critical ability is that they can see this information whenever they deem it necessary to do so for performing their jobs. There is an intuitive notion that information restricted to the higher levels of the hierarchy is of greater value to direct members of the top few groups. It is after all this information which is excluded from the rest of the hierarchy and therefore is the critical resource for the supervisory groups. In order to control the quantity and quality of such information it is reasonable to control the manner in which users can generate it. Thus it makes little sense to allow the direct members of p the ability to mark their files as being accessible by members of s_a alone. For this reason we suggest the policy that a user can mark his files as being accessible only by groups which are dominated by his own direct group. This is a reasonable policy except that it totally cuts off communication strictly upward in the hierarchy. So we modify it to allow direct members of a group to mark their files as being accessible by the immediate predecessors of that group. For example, direct members of t_1 can make their files accessible to any of s_s, s_h, t_1, or p. This policy limits the manner in which direct members of t_1 can share their files with other users, but the limits are very useful and impose a discipline adhering to the natural structure of the hierarchy. Thus members of two task teams cannot share a file without making it available to all task teams of that project. At the same time the members of a task team cannot bypass the hierarchy in making their files available to nonimmediate predecessors while withholding them from immediate ones. Thus direct members of t_4 can make their files available to direct members of s_a, s_s, or s_h only by making them accessible by direct members of s_3. So the immediate predecessors of a group cannot be bypassed in making files available strictly upward in the hierarchy.

These examples demonstrate that immediacy is a natural concept in a hierarchy which has useful practical applications for access control policies. This is particularly so in an n-tree, which is after all constructed by combining trees and inverted trees. In a tree each node, other than the root, has a unique immediate predecessor. Similarly, in an inverted tree each node, other than the root, has an unique immediate successor. An n-tree allows multiple immediate predecessors and successors, but these are the net result of incremental decisions where at each step we have uniqueness. Therefore there is strong reason for immediacy to be significant in an n-tree.

IV. The Dimension of an N-Tree

In this section we review the basic result of [12] that any n-tree can be represented as the intersection of two linear orderings. That is $u < v$ in the n-tree if and only if u precedes v in both linear orderings. An n-tree can

[1] We would like to use the term mandatory controls for such inviolable limits. However this term is often used in the narrow sense of confidentiality policies in the military based on levels and compartments, so we refrain from using it here.

therefore be represented by assigning a pair of integers $l[x]$ and $r[x]$ to each node x whose values are the position of x in the two linear orderings, respectively. To determine whether node u is a predecessor of node v we need only check whether $l[u] < l[v]$ and $r[u] < r[v]$. We refer to these numbers individually as the *l values* and *r values* and jointly as the *lr values*.

By the familiar procedure of topological sorting we know that every partial order P on a set of elements G can be extended to a linear ordering of G. In general, there will be more than one linear extension of P. Let $\Gamma(P)$ be the collection of all linear extensions of P. The intersection of linear orderings $L_1, L_2, \cdots, L_k$ is defined as the set of ordered pairs

$$\{(u, v) | (u, v) \in L_1 \wedge (u, v) \in L_2 \cdots \wedge (u, v) \in L_k\}.$$

A *realizer* of P is a subset of $\Gamma(P)$ whose intersection equals P. The *dimension* of a partial order P is the size of the smallest realizer of P. This concept was introduced by Dushnik and Miller [3]–[5].

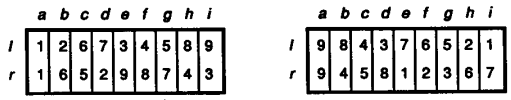
Every rooted tree has a size two realizer obtained by a left-to-right preorder traversal and a right-to-left preorder traversal. For the rooted tree of Fig. 1(d) these traversals are, respectively, *abefgcdhi* and *adihcbgfe*. This tree can be represented by assigning *lr* values as shown in Fig. 6(a). From these *lr* values it is then easy to check, for instance, that a and b are predecessors of e while c and e are incomparable. A size two realizer for an inverted rooted tree can be computed by reversing the linear orderings which comprise a size two realizer for the corresponding rooted tree. For the inverted tree of Fig. 1(e) these traversals are respectively *ihdcgfeba* and *efgbchida* leading to the *lr* values of Fig. 6(b). A proof of these observations is quite straightforward [12].

Theorem 1: A partial order whose Hasse diagram is a rooted tree or an inverted rooted tree has a realizer of size two.

Proof: It suffices to consider the case of a rooted tree. We claim that the linear orderings obtained by the left-to-right preorder and right-to-left preorder traversals of the tree constitute a realizer for the tree. If $u < v$, that is u is the root of a subtree which includes v, it is obvious that u will precede v in both preorder traversals. On the other hand if u and v are incomparable in the tree there must be some w such that $w < u$ and $w < v$, that is w is the root of a subtree which includes u and v. Without loss of generality let the path in the tree from w to u be to the left of the path from w to v. But then u will precede v in the left-to-right preorder traversal and will follow v in the right-to-left preorder traversal. □

The extension of this property to n-trees is easily shown due to the following result, first proved by Hirugachi [5], [6], that refinement does not increase dimension.

Theorem 2: Let P and Q be partial orders on disjoint sets G and H, respectively. Let $u \in G$. If P' is the partial order obtained by the refinement of u in P into Q then the dimension of P equals the bigger of dimension P or dimension Q.



(a) (b)

Fig. 6. Example of *lr* values for a tree.

It follows that if P and Q have dimension less than or equal to two then so does P'. On the basis of Theorems 1 and 2, we have the following result for n-trees [12].

Theorem 3: Every n-tree has a realizer of size two.

Proof: Because of Hirugachi's theorem we need only show that a partial order whose Hasse diagram consists of a forest of mutually disjoint rooted trees and inverted rooted trees has a realizer of size two. An empty partial order, where all distinct elements are pairwise incomparable, has a size two realizer obtained by any linear ordering of the elements and its reverse. A forest of rooted trees and inverted trees can be obtained by refining the elements of an empty partial order one at a time into a rooted tree or inverted rooted tree as appropriate. □

We conclude this section by noting that there is an efficient algorithm for computing a size two realizer for any two dimension partial order and therefore for n-trees in particular [12]. The algorithm is based on the following characterization of two-dimensional partial orders due to Dushnik and Miller [3]. The *incomparability graph* of a partial order is the undirected graph whose vertices are the vertices of the partial orders with an edge between u and v if u and v are incomparable. An undirected graph is *transitively orientable* if and only if we can assign a direction to each edge so the resulting directed graph has no cycles. Dushnik and Miller proved that the dimension of a partial order is less than or equal to two if and only if its incomparability graph is transitively orientable. An algorithm for recognizing transitively orientable graphs and assigning an orientation was presented by Pnueli, Lempel, and Even [9]. Golumbic [5] shows this algorithm has low degree polynomial complexity. A transitive orientation of the incomparability graph and it reverse orientation, along with the partial order itself give us the two linear orderings which constitute a size two realizer. This last step amounts to topological sorting of an acyclic directed graph, for which efficient algorithms are well known. Since the incomparability graph can be easily constructed in polynomial time, this entire procedure has low degree polynomial complexity.

V. Immediacy in an N-Tree

In the previous section we showed that a pair of integers $l[x]$ and $r[x]$ can be assigned to each node x in an n-tree, such that $u < v$ if and only if $l[u] < l[v]$ and $r[u] < r[v]$. We now extend this representation of an n-tree so that it is easily determined whether or not u is an immediate predecessor of v, that is whether or not there exists a node k with $u < k < v$. For this purpose, we propose to assign four additional integers to each node x as follows.

1) $l^-[x]$ is the minimum value of $l[y]$ for all immediate predecessors y of x. If x has no predecessors then $l^-[x] = l[x]$.

2) $r^-[x]$ is the minimum value of $r[y]$ for all immediate predecessors y of x. If x has no predecessors then $r^-[x] = r[x]$.

3) $l^+[x]$ is the maximum value of $l[y]$ for all immediate successors y of x. If x has no successors then $l^+[x] = l[x]$.

4) $r^+[x]$ is the maximum value of $r[y]$ for all immediate successors y of x. If x has no successors then $r^+[x] = r[x]$.

The top two rows of Fig. 7 show a possible assignment of lr values for the n-tree of Fig. 5(c). The remaining rows show the values for these additional four integers derived from the lr values.

Our claim is that node u is an immediate predecessor of node v if and only if the following inequalities hold.

$$l^-[v] \le l[u] < l[v] \le l^+[u]$$
$$r^-[v] \le r[u] < r[v] \le r^+[u]$$

The condition that $l[u]$ and $r[u]$ are, respectively, less than $l[v]$ and $r[v]$ will hold for all $u < v$. The additional conditions stated above distinguish the immediate predecessors of v from the nonimmediate predecessors. It is convenient for us to treat the l and r values as a pair denoted by $lr[u]$, that is $lr[u] = <l[u], r[u]>$. We define inequality of lr values as follows.

$$lr[u] < lr[v] \equiv l[u] < l[v] \wedge r[u] < r[v]$$
$$lr[u] \le lr[v] \equiv l[u] \le l[v] \wedge r[u] \le r[v]$$

The condition for u to be a predecessor of v can then be written as $lr[u] < lr[v]$. We similarly define $lr^-[x]$ and $lr^+[x]$ to be the pairs $<l^-[x], r^-[x]>$ and $<l^+[x], r^+[x]>$, respectively. The condition for u to be an immediate predecessor of v is then stated as follows.

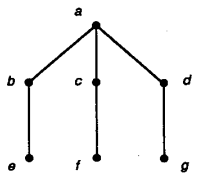
$$lr^-[v] \le lr[u] < lr[v] \le lr^+[u] \qquad (1)$$

If u is an immediate predecessor of v this condition follows trivially from the definitions so it certainly is a necessary condition. The proof that it is sufficient is the difficult part.

Before proving that our claim is true let us see if we really need both lr^- and lr^+ in condition 1. For the values shown in Fig. 7 it can be seen by inspection that either of the conditions $lr^-[v] \le lr[u] < lr[v]$ or $lr[u] < lr[v] \le lr^+[u]$ correctly identify that u is an immediate predecessor of v. So, in this case one of lr^- or lr^+ is redundant. In general however we do need both lr^- and lr^+. Consider the tree in Fig. 8 with lr, lr^-, and lr^+ values as shown. In this case the condition $lr^-[v] \le lr[u] < lr[v]$ correctly identifies immediate predecessors but the condition $lr[u] < lr[v] \le lr^+[u]$ incorrectly identifies a as an immediate predecessor of f. If we invert this tree the former condition will be incorrect while the latter will be correct.

	s_a	s_s	s_h	t_1	t_2	s_3	t_4	t_5	p_3	p
l	1	2	3	4	5	6	7	8	9	10
r	1	3	2	9	8	4	6	5	7	10
l^-	1	1	1	2	2	2	6	6	7	4
r^-	1	1	1	2	2	2	4	4	5	7
l^+	3	6	6	10	10	8	9	9	10	10
r^+	3	9	9	10	10	6	7	7	10	10

Fig. 7. lr, lr^-, and lr^+ values for the n-tree of Fig. 5(c).



	a	b	c	d	e	f	g
l	1	2	4	6	3	5	7
r	1	6	4	2	7	5	3
l^-	1	1	1	1	2	4	6
r^-	1	1	1	1	6	4	2
l^+	6	3	5	7	3	5	7
r^+	6	7	5	3	7	5	3

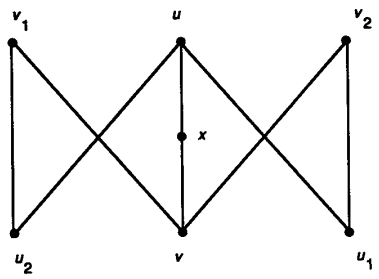
Fig. 8. Example to show that both lr^- and lr^+ are needed.

In general there will be several distinct size two realizers for an n-tree leading to different assignments of lr values. The lr values, and hence the lr^- and lr^+ values, can be based on any one of these realizers. Our claim is that condition 1 correctly determines the immediate predecessors for an lr assignment based on any size two realizer of the ntree. For the tree of Fig. 8 we can show that the condition $lr[u] < lr[v] \le lr^+[u]$ is incorrect for any lr assignment. Based on a method by Golumbic [5] for counting all transitive orientations of an undirected graph we can show there are only six distinct realizers for this tree, each one leading to a distinct lr assignment. These correspond to the six different ways that the three branches of this tree can be arranged left-to-right and computing the realizer as the left-to-right preorder and right-to-left preorder traversals. In all six cases the middle leaf will be incorrectly identified as immediate successors of the root, if we rely on lr^+ alone.

We point out that condition 1 does not work correctly for arbitrary two-dimensional partial orders. For the two-dimensional partial order of Fig. 9, condition 1 incorrectly identifies u as an immediate predecessor of v. For this case we can show by Golumbic's method for counting transitive orientations that there are only two possible lr assignments, the one shown in the figure and the other one obtained by interchanging the l and r values for each node. So condition 1 fails for every lr assignment for this partial order.

In the rest of this section we prove that condition 1 is sufficient for immediacy in n-trees. The proof is by contradiction, so we assume that condition 1 is true but u is not an immediate predecessor of v. We show this assumption implies there must be a set of nodes related as in Fig. 9. Finally we show that such a configuration cannot occur in an n-tree.

Theorem 4: For any n-tree with lr, lr^-, and lr^+ values obtained on the basis of any size two realizer, u is an



	v_1	u	v_2	x	u_2	v	u_1
l	1	2	5	4	3	6	7
r	5	2	1	4	7	6	3
l^-	1	2	5	2	1	1	2
r^-	5	2	1	2	2	1	1
l^+	6	7	7	6	3	6	7
r^+	7	7	6	6	7	6	3

Fig. 9. A two-dimensional partial order for which condition 1 is incorrect.

immediate predecessor of v if and only if $lr^-[v] \leq lr[u] < lr[v] \leq lr^+[u]$.

Proof: The only if part follows trivially from the definitions. We prove the if part by contradiction, for which purpose assume that u is not an immediate predecessor of v and $lr^-[v] \leq lr[u] < lr[v] \leq lr^+[u]$. Because u is not an immediate predecessor of v these inequalities must be strict, that is $lr^-[v] < lr[u] < lr[v] < lr^+[u]$. Also there exists an x such that $u < x < v$.

Let v_1 and v_2 be the immediate predecssors of v with minimum l and r values, respectively, that is $l^-[v] = l[v_1]$ and $r^-[v] = r[v_2]$. Similarly let u_1 and u_2 be the immediate successors of u with maximum l and r values, respectively, that is $l^+[u] = l[u_1]$ and $r^+[u] = r[u_2]$. We show that the nodes u, v, x, v_1, v_2, u_1, and u_2 must have the following relationship.

1) v_1 and v_2 are distinct incomparable predecessors of v.

Proof: v_1 and v_2 must be distinct, otherwise $lr[v_1] = lr^-[v] < lr[u]$ so $v_1 < u < v$ contrary to v_1 being an immediate predecessor of v. Since v_1 and v_2 are distinct immediate predecessors of v they are incomparable.

2) u_1 and u_2 are distinct incomparable successors of u.

Proof: Follows by a similar argument as for 1.

3) v_1 and v_2 are incomparable to u.

Proof: Since v_1 and v_2 are immediate predecessors of v neither one can be a predecessor of u. If u is a predecessor of either v_1 or v_2 we cannot have $lr^-[v] < lr[u]$.

4) u_1 and u_2 are incomparable to v.

Proof: Follows by a similar argument as for 3.

5) v_1, v_2, u_1, and u_2 are incomparable with x.

Proof: x cannot be a predecessor of v_1 and v_2, otherwise u would be a predecessor of v_1 or v_2 in contradiction to 3. Since v_1 and v_2 are immediate predecssors of v they cannot be predecessors of x. So v_1 and v_2 are incomparable to x. By a similar argument it follows that u_1 and u_2 must be incomparable to x.

6) $v_1 < u_2$ and $v_2 < u_1$.

Proof: We have $lr[v_1] < lr[u_2]$ because of $l[v_1] = l^-[v] < l[u] < l[u_2]$ and $r[v_1] < r[v] < r^+[u] = r[u_2]$. Similarly, $lr[v_2] < lr[u_1]$ because of $l[v_2] < l[v] < l^+[u] = l[u_2]$ and $r[v_2] = r^-[v] < r[u] < r[u_1]$.

7) v_1 is incomparable to u_1 and v_2 is incomparable to u_2.

Proof: To establish that v_1 and u_1 are incomparable first note that $l[v_1] < l[x] < l[u_1]$ due to $l[v_1] = l^-[v] < l[u] < l[x] < l[v] < l^+[u] = l[u_1]$. Now, from 5, we know x is incomparable to v_1 and u_1. So it must be that $r[u_1] < r[x] < r[v_1]$. But then $l[v_1] < l[u_1]$ while $r[u_1] < r[v_1]$. By a similar argument we can show that v_2 is incomparable to u_2.

We now argue that the relationships between the nodes u, v, x, u_1, u_2, v_1, and v_2 established above cannot occur in an n-tree. Let P be a partial order on the set G and let G' be any subset of G. The partial order P' obtained by *inducing* P on G' is defined as

$$P' = \{(u, v) | (u, v) \in P \wedge u, v \in G'\}.$$

A significant property of n-trees is that if P is an n-tree then any induced partial order P' must also be an n-tree [12, Theorem 5]. The seven facts established above assert that the partial order obtained by inducing the given partial order on nodes u, v, x, u_1, u_2, v_1, and v_2 is exactly as shown in Fig. 9. So if we induce the partial order on the nodes v_1, u_2, u, and v, we obtain the Hasse diagram N of Fig. 10.

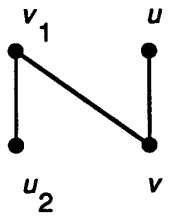


Fig. 10.

To complete the proof by contradiction we show that N is not an n-tree, so the original partial order cannot be an n-tree contrary to the theorem's statement. Now N is clearly not a tree or inverted tree. So if N is an n-tree it must be constructed by a nontrivial refinement of a node in an n-tree P into an n-tree Q, where by nontrivial we mean that both P and Q have at least 2 nodes. Let G be the set of nodes in Q and H the set of nodes in P excluding the exploded node, i.e., G and H are a partition of $\{v_1, u_2, u, v\}$. For a nontrivial refinement, $|G| \geq 2$ and $|H| \geq 1$. Combined with the fact that $|G| + |H| = 4$ it is evident that $|H|$ is 1 or 2. By the definition of refinement all nodes in H have the same relation to all nodes in G. Now H cannot consist of a single node since there is no

node in N which has the same relationship to the remaining three nodes. Similarly, H cannot be of size two since there is no pair of nodes in N which has the same relationship to the remaining two nodes. So it is not possible to construct N by a nontrivial refinement. Therefore N cannot be an n-tree. □

VI. Concluding Remarks

To summarize, the concept of n-trees was recently introduced by this author [12] and it was shown that n-trees are a natural hierarchy for the subgroup relation between protection groups. The n-tree is a useful and substantial generalization of the rooted tree and inverted rooted tree partial orders. We have shown how to develop complex n-trees incrementally in a top-down manner by successive refinement. N-trees have an efficient representation in terms of lr values assigned to each group, on the basis of which it is easily determined whether or not one group is a subgroup of another. Each l value or r value requires $\lceil \log (m) \rceil$ bits where m is the number of groups and we can represent the n-tree using $2 * \lceil \log (m) \rceil$ bits per group.

The main contribution of this paper is to extend this representation of n-trees by assigning four additional values to each group so it can be easily determined when a group g is an immediate subgroup h, that is whether or not there exists a group k such that $g < k < h$. This requires $6 * \lceil \log (m) \rceil$ bits per group. If $m = 1000$, that is there are hundreds of groups, it will take less than 8 bytes per group to represent the n-tree in this manner.

In [12] it is shown how to assign lr values to the groups in an n-tree so that after refinement of a group we need only assign lr values to the new groups introduced by refinement while the lr values assigned to nonexploded groups do not change. This technique is based on the idea of assigning a quota to each group which determines the maximum number of new groups that this group can be refined into, be it in a single step or by a sequence of refinements. This technique allows us to continue refining the n-tree with minimal disruption while the system is in operation. It would be interesting to see whether quota based technique can be integrated with the lr^- and lr^+ values we have now defined for recognizing immediacy.

Although n-trees cover a wide range of practical situations, as we have mentioned earlier, there are partial orders of practical importance which have dimension greater than two and hence cannot be n-trees. Most notably let S be a nonempty set and let 2^S be the power set of S, i.e., the set of all subsets of S, partially ordered by set inclusion. This class of partial orders arises naturally in the context of protection groups. For example, S can be the set of attributes whose subsets determine the compartments in military security policies [1], [2], [8]. Komm [7] proved that the dimension of the subset partial ordering on 2^S is $|S|$. Since this partial order can be represented using $|S|$ bits for each subset of S, the dimension approach is clearly not useful for this case. In our opinion this is the most important case excluded by n-trees. Clearly restricting the dimensions of partial orders to an upper bound of some small integer bigger than two does little to cover this case. We have moreover shown that our technique for recognizing immediacy is specific to n-trees and does not apply to arbitrary two-dimensional partial orders. So it appears that n-trees are the only useful application of dimension theory to the representation of the subgroup relation, particularly with the requirement of recognizing immediacy.

References

[1] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, no. 5, pp. 236–243, May 1976.

[2] D. E. Denning and P. J. Denning, "Data security," *ACM Comput. Surveys*, vol. 11, no. 3, pp. 227–249, Sept. 1979.

[3] B. Dushnik and E. W. Miller, "Partially ordered sets," *Amer. J. Math.*, vol. 63, pp. 600–610, 1941.

[4] P. C. Fishburn, *Interval Orders and Interval Graphs: A Study of Partially Ordered Sets*. New York: Wiley, 1985.

[5] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*. New York: Academic, 1980.

[6] T. Hirugachi, "On the dimension of orders," *Sci. Rep. Kanazawa Univ.*, vol. 1, pp. 77–94, 1951.

[7] H. Komm, "On the dimension of partially ordered sets," *Amer. J. Math.*, vol. 7, pp. 507–520, 1948.

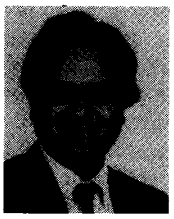
[8] C. E. Landwehr, "Formal models for computer security," *ACM Comput. Survey*, vol. 13, no. 3, pp. 247–278, Sept. 1981.

[9] A. Pnueli, A. Lempel, and S. Even, "Transitive orientation of graphs and identification of permutation graphs," *Canadian J. Math.*, vol. 23, pp. 160–175, 1971.

[10] D. M. Ritchie and K. Thompson, "The UNIX time-sharing system," *Commun. ACM*, vol. 17, no. 7, pp. 365–375, July 1974.

[11] J. H. Saltzer, "Protection and the control of information sharing in MULTICS," *Commun. ACM*, vol. 17, no. 7, pp. 388–402, July 1974.

[12] R. S. Sandhu, "The *N*-tree: A two dimension partial order for protection groups," *ACM Trans. Comput. Syst.*, vol. 6, no. 2, pp. 197–222, May 1988.



Ravinderpal S. Sandhu (M'89) received the B.Tech. degree in electrical engineering from Indian Institute of Technology, Bombay, in 1974, the M.Tech. degree in computer technology from Indian Institute of Technology, Delhi, in 1976, and the M.S. and Ph.D. degrees in computer science from Rutgers University, New Brunswick, NJ, in 1980 and 1983, respectively.

He is an Associate Professor of Information Systems and Systems Engineering at George Mason University, Fairfax, VA. Prior to that he was an Assistant Professor of Computer and Information Science at the Ohio State University. He has also held teaching and research positions at the Indian Institute of Technology, Jawaharlal Nehru University, and Hindustan Computers Limited, all in Delhi, India. His research interests include information systems security, database management systems, software engineering, distributed systems, and operating systems.