Iterated Trees of Strategies and Priority Arguments

STEFFEN LEMPP[1,3]
Department of Mathematics, University of Wisconsin
Madison, WI 53706, USA
lempp@math.wisc.edu

MANUEL LERMAN[2,3]
Department of MathematicsUniversity of Connecticut
Storrs, CT 06269-3009, USA
mlerman@math.uconn.edu

0. Introduction. It is our intent, in this paper, to try to describe some of the key ideas developed in the series or papers by Lempp and Lerman [LL1, LL2, LL3, LL4], which use the iterated trees of strategies approach to priority arguments. This is an approach which provides a framework for carrying out priority arguments at all levels of the arithmetical hierarchy. The first attempt to find a framework for (finite injury) priority arguments was presented by Sacks [Sa] in 1963, who gave the following motivation for his attempt: "Our purpose of formulating Theorem 1 of this section is to separate (insofar as possible) the combinatorial aspects of the priority method from the recursion-theoretic aspects. We do not claim that Theorem 1 stands as a fundamental principle from which all results so far obtained by the priority method follow, but we do believe that Theorem 1 and its proof will be useful to anyone who wishes to develop an intuitive understanding of the workings of the priority method in all its manifestations." Many others have presented frameworks for priority arguments or classes of priority arguments since that time, and many new ideas have changed the way we approach priority arguments. A listing of some of these attempts can be found in [LL4]. There are several key ideas which have greatly influenced the framework which is intuitively described in this paper, and we would like to point those out. The first is the *tree of strategies* approach to priority arguments which was introduced by Harrington and refined and popularized by Soare [So]; this is the way in which most recursion theorists now approach priority arguments. Groszek and Slaman [GS] and Ash [A] have developed different frameworks, and some of their ideas and ways of thinking have been incorporated into our approach.

The origin of our work was the proof of a particular theorem which required priority arguments at all levels of the arithmetical hierarchy. (As with Ash's method, there

1 Partially supported by National Science Foundation Grant DMS91-00114.

2 Partially supported by National Science Foundation Grant DMS92-00539.

3 Dedicated to Gerald E. Sacks on the occasion of his sixtieth birthday.

should be an extension to all levels of the hyperarithmetical hierarchy, but we have not tried to carry this out). We saw, along the way, that our approach was much more generally applicable, and decided to try to develop the method. (We hope to develop a detailed general presentation in a forthcoming monograph.) We set the following goals for measuring the success of the approach:

- Constructive "trees of strategies" approach.
- Applicability to all priority arguments (for r.e. degrees).
- Isolation of the combinatorial lemmas which are part of the priority method, rather than the individual proofs.
- Helpfulness in finding new proofs.
- Shortening of existing proofs.

The "trees of strategies" approach was taken so that the presentation would follow, as closely as possible, the current presentation of most arguments at lower levels of the arithmetical hierarchy. The difference is that, rather than using a single tree, we introduce a tree of strategies for those levels of the arithmetical hierarchy, beginning at the recursive level, and continuing up to the natural level for the particular theorem to be proved. This approach seems to be generally applicable, at least for priority arguments in the r.e. degrees or the lattice of r.e. sets which correspond to effective Cohen forcing, and we have carried out or sketched proofs of many theorems using this approach. We plan to look at priority arguments which effectivize other types of forcing in the future.

There is a very general list of properties which need to be verified for any given construction, and the proof that the verification of these properties suffices seems to be the common core of priority arguments. This portion of the combinatorics of priority arguments can be generally isolated. However, the verification of these properties can differ with the theorem to be proved. We have therefore taken a modular approach, finding additional properties which can be imposed, simultaneously, on many constructions, and verify the original properties from the stronger properties. This approach also leads to a better understanding of some of the techniques used *within* priority arguments, and points to similarities between proofs which were not previously noted. The method has been successfully applied to prove the following new theorems in [LL4, LL2]:

THEOREM: The existential theory of the r.e. degrees with predicates $\leq_n$, for all n, interpreting $\mathbf{a}^{(n)} \leq \mathbf{b}^{(n)}$, is decidable. ■

THEOREM: There are r.e. degrees **a**,**b** such that **a** and **b** form a minimal pair, and $\mathbf{a}'$ and $\mathbf{b}'$ form a minimal pair over $\mathbf{0}'$. ■

There is an interplay between the savings in length produced by the combinatorial lemmas, and the need to add steps to describe the decomposition of requirements from tree to tree. The latter is normally easier to carry out, but in simpler proofs, may take longer. The framework presents real advantages when the construction needs to deal with similar requirements at different levels, e.g., $\mathbf{a}' \leq \mathbf{b}'$, $\mathbf{a}'' \leq \mathbf{b}''$, etc.

It is not our intent to present a detailed development of the "iterated trees of strategies" approach in this paper, but rather, to present a broader idea of the components of this approach and the roles played by each of these components. The intuition behind the framework tends to get lost in the detail of using the framework to prove individual theorems, and certainly, the intuition obtained in developing the framework does not come across in the presentation of the finished product. It is our hope that the presentation of this overview and intuition will make the framework more intelligible to readers of papers where it is used. We will try to illustrate each idea introduced with an example, choosing particular familiar requirements where the notion under consideration is used.

1. A broad overview before modularization. Each requirement has a natural level or *dimension* based on the quantifier complexity of the statement of the requirement. In the standard terminology, finite injury or $\mathbf{0}'$ requirements have dimension 1, the standard infinite injury or $\mathbf{0}''$ requirements have dimension 2, etc. In order to determine where to carry out an iterated trees of strategies argument, we let n be the largest dimension of a requirement which must be satisfied, and carry out the proof simultaneously on the trees $T^0,\ldots,T^n$. We begin by assigning all requirements, irrespective of their dimensions, to nodes of T^n. This assignment must be done so that satisfaction of all requirements along any path through T^n will ensure the theorem being proved. (It is sufficient to have this property only for those paths through T^n which are computed by a possible construction on T^0.) Following the assignment, a level-by-level decomposition process is specified; having determined the roles of nodes of T^{k+1}, the decomposition process assigns, to nodes of T^k, subrequirements of requirements assigned to nodes of T^{k+1}, together with information about how and when they are expected to act. This is carried out through an assignment of a node η on T^k to a node σ of T^{k+1} for which η works; η is called a *derivative* of σ. The decomposition continues down to the recursive level, namely, to T^0. The construction

associated with a proof takes place along a recursive path Λ^0 through T^0, and action is determined by the information associated with nodes of T^0. We then have a path computation function which computes a path on T^{k+1} from one on T^k, and must essentially act as the inverse of the decomposition process; by this we mean that the satisfaction of the instructions assigned to *critical nodes* on a path $\Lambda^k \in [T^k]$ must ensure the satisfaction of the instructions assigned to *critical nodes* on the path $\Lambda^{k+1} \in [T^{k+1}]$ computed by Λ^k. (The reader should think of critical nodes as those nodes lying along the true path which ensure the ultimate satisfaction of requirements.)

More specifically, the processes described above work as follows. A requirement supplies a node of a tree with several pieces of information; a *directing sentence* indicating when the node should take action; another sentence indicating the action to be taken by the node if the directing sentence is false (this is called the *activated action*); and a third sentence indicating the action to be taken by the node if the directing sentence is true (this is called the *validated action*). This information must be passed down from tree to tree, usually by bounding quantifiers. A *weight function* wt is defined on nodes of trees, and is used to determine the bounding parameters and numbers on which requirements act. We demonstrate what these sentences are in some familiar cases.

Example 1.1: Friedberg-Muchnik Incomparability Requirements: In the Friedberg-Muchnik Theorem [F],[M], r.e. sets A and B of incomparable Turing degree are constructed. A typical requirement assigned to a node σ of T^1 has the form $\Phi(A) \neq B$, where Φ is a partial recursive functional. An argument $x = wt(\sigma)$ is specified for this requirement. The *directing sentence* is $\Phi(A;x)\downarrow = 0$, *activated action* is $B(x) = 0$, and *validated action* is $B(x) = 1$. If the directing sentence is false and activating action is followed, then $B(x) = 0$ and either $\Phi(A;x)\uparrow$, or $\Phi(A;x)\downarrow \neq 0$; and if the directing sentence is true and validated action is followed, then $B(x) = 1$ and $\Phi(A;x)\downarrow = 0$. In either case, the requirement is satisfied.

Suppose that the decomposition process determines that the node $\eta \in T^0$ is to work to satisfy the requirement R assigned to σ. The framework will then use $s = wt(\eta)$ as a *stage* of the construction. The *directing sentence* for η is $\Phi_s(A^s;x)\downarrow = 0$, where Φ_s and A^s are the approximations to Φ and A, respectively, at stage s. *Activated action* is $B^{s+1}(x) = 0$, and *validated action* is $B^{s+1}(x) = 1$, where B^{s+1} is the approximation to B at stage s+1. If the directing sentence for η is false, then a node $\xi \supset \eta$ will be assigned to work for σ on a stage $t = wt(\xi) > s$. If all such directing sentences are false, then R will be satisfied. If the directing sentence assigned to η is true and validated action is followed, then R will be satisfied as long as the oracle information used from A^s is not changed; and this is ensured by the way the framework will assign numbers used by the various requirements. ■

Example 1.2: Thickness Requirements. Let Q be an infinite recursive set, and let S(x,y) be a recursive predicate. We want to build an r.e. set A such that:

$$\forall x \exists y S(x,y) \rightarrow Q \subseteq^* A, \text{ and}$$

$$\neg \forall x \exists y S(x,y) \rightarrow Q \cap A =^* \emptyset$$

where $=^*$ means "differs finitely from" and $\subseteq^*$ means "has only finitely many elements not in". Suppose that this requirement is assigned to a node $\zeta \in T^2$. The *directing sentence* for ζ is $\forall x \exists y S(x,y)$, *activated action* is $Q \cap A =^* \emptyset$, and *validated action* is $Q \subseteq^* A$. Let σ be a derivative of ζ on T^1. A number $u = wt(\sigma)$ will be assigned to σ. The *directing sentence* for σ is $\forall x \leq u \exists y S(x,y)$, *activated action* is $Q \subseteq Q \cap [0,u)$, and *validated action* is $p \in A$ for all $p \geq u$ such that $p \in Q$ and p is *controlled* by σ. One should think of control as designating those p for which σ has the responsibility to determine placement into A. If the directing sentence for σ is false, then σ will control all such $p \geq u$, and will prevent all such p from entering A. If the directing sentence for σ is true and both ζ and σ lie along the true paths of their respective trees, then a shortest derivative τ of ζ extending σ will appear on the true path through T^1, and σ will control the numbers in $[wt(\sigma),wt(\tau)) \cap Q$. It then follows that the correctness of directing sentences and action is carried up from T^1 to T^2. If η is a derivative of σ on T^0, then $s = wt(\eta)$ provides a stage at which we check to see the truth of the directing sentence for σ, namely $\forall x \leq u \exists y \leq s S(x,y)$. The decomposition from T^1 to T^0, and specification of directing sentences and action, is similar to that described in Example 1.1, so we do not repeat that description. ■

We have tried to give a very general description of the assignment and decomposition processes by example. These must satisfy certain basic properties which are described later in this section. In the two examples presented above, the assignment and decomposition processes can be presented in a canonical way. The ordering of requirements assigned is not important, nor is the placement of derivatives in the decomposition process, as long as certain basic properties are satisfied. This is not the case in general, as we will see with later examples. Thus for each construction, an assignment must be specified and shown to satisfy the basic properties. Modularization can be used here so that satisfaction of the basic properties can be ensured if certain other properties (easily verified in the given situation) are satisfied.

Having carried out the assignment and decomposition processes, a construction will now take place on T^0, and will be based, almost exclusively, on the directing sentences and the action specified by the nodes which are encountered. In some cases, some of these nodes will have forced action, and the sentences and action associated with the nodes will

be ignored; if such a node lies on the true path, it will act consistently with the desires of other requirements on the true path. Once the construction is complete, we will have to prove the theorem by climbing back up, tree by tree, to T^n, and relating the action taken at one level to the action taken at the next level. The path directing function λ will provide a way to approximate, in the limit, to a path through T^{k+1} from a path through T^k. The reader may have noted that we have yet to specify any details about the trees; it is in the definition of λ that the definition of the trees becomes important.

The standard one-tree models of priority arguments provide misleading examples of how to define trees. Finite injury arguments require only binary trees. While the standard infinite injury model uses infinitely branching trees, the nodes corresponding to a fixed requirement R normally do not carry more information than the facts that different nodes correspond to different attempts at satisfying R, and the order in which the attempts at satisfaction are made. One can get by with this information for T^3 as well. But at higher levels, this method of coding nodes loses too much information, precluding a close interaction between the decomposition process and the path computation process. Priority characterization in terms of a node being to the left or right of another also disappears at higher levels, and is replaced by the decomposition machinery.

In our presentation, $T^0 = \{0,\infty\}^{<\omega}$, with the outcome 0 corresponding to activated action, and the outcome ∞ corresponding to validated action. For the other trees, we will want a node of T^{k+1} to uniquely recover the construction followed to reach that node. In order to accomplish this, we want the outcome for each node to contain the information telling us how that outcome was obtained. Suppose that $\sigma \in T^{k+1}$ is given, and let ρ be its immediate predecessor. For definiteness, suppose that the directing sentence for ρ begins with an unbounded block of universal quantifiers. (If the quantifiers are existential, proceed similarly with the negation of the directing sentence, interchanging activated and validated action.) If σ codes the fact that we follow the activated action for ρ, then we will want there to be be a uniform bound on each quantifier in the block, yielding a sentence which is false. This should be discovered at a derivative ξ of ρ on T^k, where the corresponding directed sentence has such a bound, and so ξ will have an immediate successor η along this true path indicating that activated action was followed. In this case, we want $\lambda(\eta) = \sigma$; thus σ knows the location of the node ξ determining the truth or falsity of the directing sentence for ρ, and the outcome at that location, indicating the truth or falsity of the sentence. The trees do not need derivatives of ρ which extend η. Now suppose that σ codes the fact that we follow the validated action for ρ. Then for every bound on the leading block of quantifiers in the directing sentence for ρ, the induced sentence is true. σ must then code the fact that this is the case. There will then have to be infinitely many derivatives of ρ on the true path through T^k, each having validated outcome

along this true path. σ codes this by determining that the outcome of ρ along σ is the first node along the corresponding path through T^k which is a derivative of ρ, together with the validated outcome for this node. Thus each tree is a tree of finite sequences of nodes from the previous tree; the last such node in σ carries the guess of the current approximation to the true path of T^k about the way ρ will be satisfied. It seems to be impossible to get the framework machinery to work unless this information is coded into the paths. With this in mind, the framework theorem implies that all requirements are satisfied as long as we can verify the following:

(1.1) The assignment process of requirements to nodes of T^n is recursive, and for each possible true path through T^n, the satisfaction of all requirements along that path ensures the satisfaction of the theorem being proved.

(1.2) The decomposition process from T^{k+1} to T^k is recursive. Furthermore, if $\Lambda^k \in [T^k]$ is a possible true path through T^k then $\lambda(\Lambda^k) = \Lambda^{k+1}$ is infinite. If $\sigma \subset \Lambda^{k+1}$ requires infinitely many derivatives, then σ has infinitely many derivatives along Λ^k; otherwise, σ has a last derivative along Λ^k which determines its outcome.

(1.3) If $\Lambda^0 \in T^0$ is the true path for the recursive construction, then for every critical node $\eta \subset \Lambda^0$, Λ^0 extends the validated outcome of η and the construction carries out the validated action for η if the directing sentence for η is true; and Λ^0 extends the activated outcome of η and the construction carries out the activated action for η if the directing sentence for η is false. Furthermore, for such η, the truth of the directing sentence for η or the action taken is not injured by action taken for longer nodes along Λ^0.

(1.4) For all $k < n$, if Λ^k is the true path through T^k and $\Lambda^{k+1} = \lambda(\Lambda^k)$, then the correctness of the prediction by Λ^{k+1} about the truth of the directing sentence for $\sigma \subset \Lambda^{k+1}$ is ensured by the correctness of the predictions by Λ^k about the truth of the directing sentences for derivatives of σ.

(1.5) For all $k < n$, if Λ^k is the true path through T^k and $\Lambda^{k+1} = \lambda(\Lambda^k)$, then the correctness of the action along Λ^{k+1} for $\sigma \subset \Lambda^{k+1}$ is ensured by the correctness of the action along Λ^k for the derivatives of σ along Λ^k.

2. Limitations on canonical processes. For some priority method proofs, one can follow a canonical process for the assignment and decomposition of requirements, and all nodes which come from the true path are critical. In this section, we present some examples to show that, in general, more care needs to be taken. We begin with a discussion of the Sacks Splitting Theorem [Sa], and assume that the reader has some familiarity with the standard proof.

Example 2.1: In order to avoid the use of trees with finite maximal paths, we restate the Sacks Splitting Theorem as follows: For every r.e. set A, there is a splitting of A into two r.e. sets, A_0 and A_1 such that either A is recursive, or each of A_0 and A_1 has strictly smaller degree than A. There are two types or requirements which need to be satisfied:

$$P_e\colon\ e \in A \rightarrow e \in A_0 \text{ or } e \in A_1, \text{ and } A_0 \cap A_1 = \emptyset.$$

$$N_{\Phi,i}\colon\ \Phi(A_i) = A \rightarrow f = A.$$

where Φ is a given partial recursive functional, and f is a recursive function being constructed. P_e has dimension 1, while $N_{\Phi,i}$ has dimension 2.

We begin with a description of the satisfaction of $N_{\Phi,i}$. Suppose that $N_{\Phi,i}$ is assigned to a node $\zeta \in T^2$. The directing sentence for ζ is $\Phi(A_i) = A$, and there is no activated action for ζ. Validated action for ζ is to define a recursive function f which is the characteristic function of A. We note that if $N_{\Phi,i}$ has validated outcome along the true path of T^2, then no requirement $N_{\Psi,j}$ is assigned to any node extending a validated outcome of ζ. (The canonical assignment would be to assign a given requirement to each node on a fixed level of T^2.) The absence of requirements $N_{\Psi,j}$ on such paths allows us to arrange for (1.1)-(1.5) to be satisfied.

A derivative σ of ζ on T^1 will have directing sentence $\Phi(A_i;x) = A(x)$ for all $x \leq z = wt(\sigma)$. Again there is no activating action, and validated action is to define $f(x) = A(x)$ for all $x \leq z$. A derivative ξ of σ on T^0 operates at stage $s = wt(\xi)$ in the same way as in Example 1.1. We note that if ξ is validated along the true path of T^0 and η is the immediate successor of ξ along this path, then σ has outcome η along the true path of T^1, and the maximum use u of the computations $\Phi_s(A_i^s;x)$ for $x \leq z$ is s which will be $< wt(\eta)$. In order for the construction to satisfy (1.4), we must restrain $A_i^s|`u+1$.

Now consider the requirement P_e. The standard constructions do not place these requirements on the tree, but treat them from outside the tree. This is an ad hoc solution, which is difficult to incorporate into generalizations. Our treatment of these requirements on the trees is identical to the ad hoc procedure. Thus we assign this requirement a to node χ along any given path through T^2, but in the decomposition process, as the dimension of

P_e is 1, χ will have at most one derivative τ along any path through T^1. Furthermore, the directing sentence and action will depend on τ, rather than on χ. Decomposition from T^1 to T^0 will be the stage decomposition of Example 1.1, and will be carried out in the obvious way.

Fix τ. The directing sentence for τ will be $e \in A$, and activating action is $e \notin A_0$ and $e \notin A_1$. As a construction is deterministic, the validating action will have to be one of the following: $e \in A_0$; or $e \in A_1$. The choice made by Sacks externally needs to be specified internally, and depends on the location of τ on T^1. This is where a particular decomposition process is important. Suppose that a requirement $N_{\Phi,i}$ is assigned to a derivative σ on T^1 which has validated outcome along the computed path, and let ρ be the immediate successor of σ along this path. Recall that ρ sees the need to restrain $A_i^s|`u+1$. We now ensure this restraint by assigning all requirements P_m for $m \leq u$ along extensions of ρ if they have not yet been assigned along σ; and this assignment precedes the assignment of any requirement $N_{\Psi,j}$ to a node which extends ρ. For each node working for such a P_m, the validating sentence is $m \in A_{1-i}$. (Note that this duplicates the external action imposed by Sacks.) If there is no $N_{\Phi,i}$ imposing such restrictions on τ, then we arbitrarily set validating action as $m \in A_0$. We have thus essentially duplicated the Sacks construction on the iterated trees of strategies, but note that this required a decomposition process highly dependent on the particular requirements being satisfied. ■

The minimal pair construction of Lachlan [L1] and Yates [Y] is a more complex example of a construction where the decomposition process is requirement-dependent. We refer the reader to [LL2] for a proof of the minimal pair theorem using iterated trees of strategies.

There are two situations which we have identified where it is important to restrict (1.3) to critical nodes. One situation deals with implication chains, and is discussed in a later section. The other situation is when the satisfaction of a requirement uses a basic module having more than one element. In this case, only the terminal nodes of the basic module along the true path are critical; the interior nodes perform action which may be injured by later nodes of the module. A simple example of such a proof is the construction of a proper d-r.e. degree. Such degrees were originally constructed by Cooper [C], but a much simpler proof was given by Lachlan (see Epstein [E]). Hinman [H] carried out this proof in the iterated trees of strategies format. We indicate the main ideas in the next example.

Example 2.2: A typical requirement for the construction of a d-r.e. set D which is not of r.e. degree is:

$$\exists u(\Phi(W|\grave{}u;x) \neq D(x) \text{ or } \exists y<u(\Psi(D;y) \neq W(y)))$$

where W is an r.e. set and Φ and Ψ are partial recursive functionals. The basic module for such a requirement R on T^1 is assigned to two consecutive nodes $\sigma \subset \tau$ along various paths through T^1. The directing sentence for σ is

$$\exists u(\Phi(W|\grave{}u;x) = 0 \ \& \ \forall y<u(\Psi(D;y) = W(y))),$$

activated action is D(x) = 0, and validated action is D(x) = 1, where x = wt(σ). The sentence is carried down to a derivative ξ of σ on T^0 by bounding u with wt(ξ), and asking for the truth of the sentence at stage wt(ξ), i.e., replacing all sets and functionals with there approximations at stage wt(ξ). σ will be a *critical node* along any $\rho \supset \sigma$ such that σ is activated along ρ; along such paths, the directing sentence will be false, and the reader can check that if activating action is taken, then the requirement will be satisfied. If the construction follows the validated action, however, we cannot protect the satisfaction of the requirement; it is possible for W|`u to change, and as we have changed D by setting D(x) = 1, we will not be imposing a difference between $\Psi$(D)|`u and W|`u. Thus it may be the case that the directing sentence for σ is false at the end of the construction. However, σ has relinquished the responsibility for satisfying R to τ. The directing sentence for τ is $\exists z<u(W(z) \neq W^{wt(\xi)}(z))$. (Note that τ can recover ξ, as η will be the outcome of σ along τ and ξ will be the immediate predecessor of η; hence we have a uniform recursive method of obtaining ξ from τ.) Activated action is D(x) = 1, which suffices to satisfy the requirement if the directing sentence for τ is false, for it will then be the case that $\Phi(W;x) = \Phi_{wt(\xi)}(W^{wt(\xi)};x) = 0 \neq 1 = D(x)$. If, however, we find this sentence to be satisfied at a derivative $\eta \in T^0$ of τ, then it will be the case that for some z < u, $W^{wt(\xi)}(z) = 0$ and $W^{wt(\eta)}(z) = 1$. We specify D(x) = 0 as validated action; R will then be satisfied as automatic restraint generated by the framework will ensure that $\Psi(D;z) = \Psi_{wt(\xi)}(D^{wt(\xi)};z) = W^{wt(\xi)}(z)$ for all z < u, while $W(z) \neq W^{wt(\xi)}(z)$ for some z < u. τ is a critical node along all nodes properly extending τ, and its action and the truth of its directing sentence can be preserved. ■

Basic modules are used when the directing sentence is a proper disjunction. The possible disjuncts are ordered, and we try to satisfy them in order, with the failure of one leading to an attempt to satisfy the next. When action is described as a disjunction, then this will frequently lead to nonuniformity of the decomposition process.

3. Decomposition Principles. We have listed (1.1)-(1.5) as the properties which need to be satisfied to carry out a proof within the Lempp-Lerman framework. But beyond the definition of our trees and the description of the path computation function, we have not yet indicated what combinatorial principles can be proved, or how to carry out the various steps. There are certain restrictions on the decomposition process, which we now discuss, which will enable us to prove lemmas of this type. These are applicable in all cases which we have investigated. Without these restrictions, we were unable to get a nice relationship between decomposition, action, and path computation which would enable us to prove general combinatorial lemmas.

We have referred to the *true path* for the construction without defining what we mean; the term is standard in the single tree case, and its generalization is the obvious one. A construction determines a recursive path Λ^0 through T^0. The path generating function λ now inductively determines a path Λ^{k+1} through T^{k+1} from a path Λ^k through T^k via a limit process, and we write $\Lambda^{k+1} = \lambda(\Lambda^k)$. For all k, Λ^k is the *true path* through T^k. Similarly, λ can be used to generate path approximations. If $\eta^k \in T^k$, then we inductively define $\eta^{k+1} = \lambda(\eta^k)$, and call η^{k+1} the *current path* through T^{k+1} at η^k. Given $r > k$, $\sigma^r \in T^r$, and $\sigma^k \in T^k$, we call σ^r an *antiderivative* of σ^k if σ^k is (hereditarily) a derivative of σ^r.

The first restriction which we place on the assignment process for a construction beginning on T^n is:

(3.1) If $k < n$ and we specify that $\sigma^k \in T^k$ is a derivative of $\sigma^{k+1} \in T^{k+1}$, then all antiderivatives of σ^{k+1} are on the current path at σ^k.

Another very important restriction (which implies (3.1), although this implication is not immediately apparent) is that, under the hypothesis of (3.1), there should be no links restraining antiderivatives of σ^{k+1}. Links were introduced by Lachlan [L2] in a $\mathbf{0}'''$-injury setting, but are needed in priority arguments at all levels; at small levels, however, it is easy to incorporate the links into the natural assignment and action description, so that they are not noticed. A *primary link* along η connects two nodes $\xi,\delta \subset \eta$ which have the same antiderivative, but one has activated outcome along η, and the other has validated outcome along η. Primary links are generated from smaller levels (except on T^0), so can be pulled down to lower level trees in a natural way to links which are not primary. The corresponding restriction is:

(3.2) If $k < n$ and we specify that $\sigma^k \in T^k$ is a derivative of $\sigma^{k+1} \in T^{k+1}$, then no antiderivative of σ^{k+1} is restrained by a link along a current path computed by σ^{k+1}.

The other restrictions have been mentioned earlier. A node σ^{k+1} will have at most one derivative σ^k along a given path until we are at a level below the dimension of the requirement assigned to σ^{k+1}. And once we are at such a smaller level, no derivatives of σ^{k+1} can extend a derivative σ^k of σ^{k+1} whose outcome indicates that a witness has been found for the directing sentence for σ^{k+1} when that sentence begins with an unbounded existential quantifier (or for the negation of the directing sentence for σ^{k+1} when that sentence begins with a universal quantifier).

With the above restrictions in place, we can prove a very crucial lemma which allows us to pass easily by induction from level to level, and to show that, if we have an assignment process which yields infinitely many blocks (so is renewed infinitely often), then we have sufficiently many derivatives of nodes along the true path to localize the proofs of (1.4) and (1.5). This crucial lemma states that if, during the construction, we assign an outcome to a node η which switches the outcome of an antiderivative of η, then even after the switch, no such antiderivative will be restrained by a link.

4. Implication Chains. There are many constructions whose descriptions involve the action of nodes which do not lie on the current path, but have higher priority than the longest node on the current path. Constructions involving *permitting* frequently have this property, as do constructions where nodes on many paths are trying to satisfy the same (global) requirement by defining axioms on the same argument. In the latter case, when a node σ on the current path wants to act, it is sometimes necessary to allow a higher priority node τ which does not lie on the current path to act in its place.

As mentioned in the previous section, allowing nodes to act when they do not lie along the current path does not mesh well with the path computation function. Let σ and τ be as in the preceding paragraph. Our approach is to show that there is a way, called *backtracking,* to return τ to the current path which causes minimum damage, or injury, to other requirements, and we can recursively specify this way. The nodes used to accomplish the change of the current path computation have forced outcomes, consistent with what is wanted by the path on which τ lies. The backtracking principle can be summarized as follows:

Backtracking Principle: We ask whether the way of returning τ to the current path involving minimum injury will injure the apparent truth of the directing sentence for σ. If the answer is "no", then we return τ to the true path, and then let the action for τ take place. If injury to the directing sentence for σ will occur, then we will need to show that all action

taken by σ is *corrected,* i.e., we have the freedom to allow τ to act, should τ ever be returned to the current path.

The mechanism used to carry this out was introduced in [LL4], and uses implication chains. The machinery is very complex, but allows us to prove theorems which yield the principle in the preceding paragraph. When applied, there is a simple property to be checked to see if correction occurs. If the property holds, then the success of the construction using backtracking is ensured by the combinatorial lemmas proved about backtracking. If the property fails to hold, then the construction will not be successful in any formulation.

We illustrate part of the idea behind backtracking in the next example.

Example 4.1: Satisfaction of requirements of the form $A'' \leq_T B''$.

We fix an A-recursive predicate R^A coding A'', and build a recursive functional Δ satisfying, for all x the requirement R_x:

$$(\forall t \exists u R^A(x,t,u) \rightarrow \lim_t \lim_u \Delta(B;x,t,u) = 1) \;\&$$
$$(\exists t \forall u \neg R^A(x,t,u) \rightarrow \lim_t \lim_u \Delta(B;x,t,u) = 0).$$

Suppose that this requirement is assigned to a node ζ of T^2. The directing sentence for ζ will be $\forall t \exists u R^A(x,t,u)$, activated action will be $\lim_t \lim_u \Delta(B;x,t,u) = 0$, and validated action will be $\lim_t \lim_u \Delta(B;x,t,u) = 1$. If $\sigma \in T^1$ is a derivative of ζ, then the directing sentence for σ will be $\forall t \leq wt(\sigma) \exists u R^A(x,t,u)$, activated action will be $\lim_u \Delta(B;x,t,u) = 0$ for all t controlled by σ along the current path (see the discussion of thickness requirements in Example 1.2 for an idea about the notion of control), and validated action will be $\lim_u \Delta(B;x,t,u) = 0$ for all t controlled by σ along the current path. If $\xi \in T^0$ is a derivative of σ, then the directing sentence for σ will be $\forall t \leq wt(\sigma) \exists u \leq wt(\xi) R^A(x,t,u)$, activated action will be $\Delta(B;x,t,u) = 0$ for all $\langle t,u \rangle$ controlled by ξ along the current path through T^0, and validated action will be $\Delta(B;x,t,u) = 1$ for all $\langle t,u \rangle$ controlled by ξ along the current path through T^0.

Now consider two incomparable nodes $\zeta, \kappa \in T^2$ to which R_x is assigned. Suppose that $\sigma \subset \rho$ are comparable nodes of T^1, σ is a derivative of ζ, ρ is a derivative of κ, ρ and κ lie on the current path but ζ does not, and the directing sentence for ρ is true at some derivative ξ of ρ. It will be the case that $wt(\rho) > wt(\sigma)$, so the directing sentence for σ will also be true, i.e., the directing sentence for ρ will imply the directing sentence for σ, allowing us to establish an implication chain. We will act to validate ξ, but first check to

see if the process of bringing ζ back to the current path will injure the truth of the directing sentence. This process must switch the outcome of the longest δ which is extended by both ζ and κ, and possibly some other nodes, and does so by switching outcomes of certain nodes on T^1 from activated to validated; one node of T^1 which is switched is a derivative of δ. When nodes of T^1 are switched in this way, we must follow their validated action, and thereby put numbers into sets, in order to later show that (1.4) and (1.5) are satisfied. As nodes of T^1 of dimension 1 can be switched only once on T^2, it will suffice to consider only the effects of switching nodes of dimension ≥ 2; in fact, by the way control and implication chains interact for requirements of this type, it will suffice to consider only the effects of switching nodes of dimension ≥ 3. For such nodes, we require constructions to have the property that whenever specified action causes a number to be placed into A, it will also cause a number of similar magnitude to be placed into every B with $B'' \geq_T A''$. Thus an injury to the oracle for the directing set for ρ will allow correction of the action prescribed by ρ. If no injury to the directing sentence for ρ will occur in backtracking to σ, then we carry out the backtracking process, the directing sentence for σ will also be true, and we validate σ. Otherwise, if ζ ever returns to the current path, then correction of axioms defined by ρ will have taken place, so σ can carry out the action prescribed. ■

In general, implication chains need to be carried down, level by level, through a complex alternating machinery. We refer the reader to [LL4] for a detailed description.

5. Control and correction. For some types of requirements R, the determination of the node which has the responsibility for specifying an axiom or placing an element into a set is clear. Such is the case in Example 1.1, where each node σ of T^1 to which a diagonalization requirement is assigned has the responsibility for $x = wt(\sigma)$. For the thickness requirements of Example 1.2, or for the double jump comparability requirements of Example 4.1, this is no longer the case. For such requirements, the determination of control of axioms is very delicate, as the need to compute limits requires the correction of axioms specified by the wrong node. A delicate generalization of the scheme mentioned in Example 1.1 can be used to determine control, subject to correction properties of other requirements. This generalization uses the implication chains mentioned in the preceding section to handle correction and existence of limits when control is exchanged between nodes with different antiderivatives on T^n (the highest level for the construction) at a given stage of a construction. A delicate level-by-level decomposition of control using the weight function allows the proof of a combinatorial lemma, which implies that when control is

exchanged on T^1, then a node of dimension at least as high as the dimension of R has its outcome switched from activated to validated. In the case of diagonalization requirements where original control is localized to a single node of dimension dim(R), the node which is validated will have an antiderivative which is also an antiderivative of the node gaining control. As validation will place an element in the oracle set for the axiom being newly controlled, correction will be possible.

Highness requirements and their generalizations ($high_2$-ness requirements, etc.) are treated similarly. A $high_n$-ness requirement will have dimension n+1, and will require specification of axioms from an oracle A which is $high_n$. One way of ensuring the satisfaction of such a requirement is to require all nodes of dimension $\geq$ n+1 to place elements into sets, and whenever an element is placed into B, to require an element of similar size to be placed into all C such that $C^{(n)} \geq_T B^{(n)}$. This will ensure that such an element will be placed into A, thus allowing the correction of axioms.

6. Configurations. The term *configuration* was introduced by Slaman [Sl] to describe a certain type of construction. Constructions requiring the building of configurations had been carried out much earlier, but Slaman named the combinatorial property which was common to such constructions. We have not, as yet, used our framework to carry out constructions which require configurations on high dimensional trees, and it is possible that combinatorial lemmas will have to be proved to deal with this situation. Our current guess, based on observations of configurations at low dimensions, is that they are handled through the assignment process, together with a delegation of responsibility for action from one node to another. We view our description of the Sacks Splitting construction (Example 2.1), as an example of a situation where passive configurations need to be built. A fairly simple example of a construction where an active configuration is built is the embedding of the lattice N_5 (see Figure 6.1) into the r.e. degrees, preserving least element.

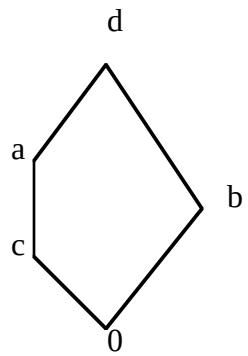

Figure 6.1

Example 6.1: N_5 is embedded into the r.e. degrees by building r.e. sets A, B, and C whose degrees **a**, **b**, and **c** have the properties of their lightface counterparts in Figure 6.1. The need for configurations comes from the interaction of the three types of requirements P_Φ: $\Phi(C;x) \neq A(x)$; and J: $\forall y(\Delta(C \oplus B;y) = A(y))$; and $N_{\Psi,\Xi}$: $\Psi(B) = \Xi(A)$ total $\rightarrow \Psi(B) = \Gamma(\emptyset)$, where we define Δ and Γ, and Φ, Ψ, and Ξ are given. Suppose that we have fixed x as above. As long as $\Phi(C;x)\uparrow$ or $\Phi(C;x)\downarrow \neq 0$, we keep x out of A. While this occurs, we will have a node whose requirement is $\Delta(C \oplus B;x) = A(x)$ so we will have to specify an axiom of the form $\Delta_s(C^s\restriction u \oplus B^s\restriction u;x) = 0$ for some s and u. And we will specify a $p < u$ to be placed into either B or C if we ever later find the need to place x into A, and so correct the above axiom. Now the minimal pair requirement $N_{\Psi,\Xi}$ does not allow us to place numbers of similar size into A and B simultaneously; and the directing sentence for P_Φ will be destroyed if we place p into C when we place x into A. So when we find that $\Phi(C;x)\downarrow = 0$, we cannot place x into A; for J would then require us to place p into either B or C, injuring either the directing sentence for P_Φ, or the action for $N_{\Psi,\Xi}$.

We build a configuration by having P_Φ delegate its permission to act to J, which acts by placing p into B, and specifying a new axiom with use > some q which is now targeted for C and is larger than the use for $\Phi(C;x)$. The placement of p into B will switch the current path (as the length of agreement measure for $N_{\Psi,\Xi}$ may not have its maximum value locally for some node along the current path), so the node σ trying to satisfy P_Φ may no longer have all its antiderivatives on the new current path. However, if this is later the case, then we are safe to let J place q into C and let P_Φ place x into A simultaneously.

The above description has not discussed the use of the framework in detail. As we have not settled on a combinatorial representation of configurations yet, we avoided pinning down the precise operation, which is not difficult to carry out in this case. The key point is that the building of configurations corresponds to having a high priority node force lower priority nodes to act in certain ways until it is safe for the higher priority node to act. ■

The preceding example described a one-step configuration. The embedding of M_5 (Figure 6.2) into the r.e. degrees requires an iteration of this process. The number of times the iteration is carried out is determined by the outcome of the node on the true path of T^3 which satisfies the requirement, and is not determined in advance. Also, nodes to which responsibility to act has been delegated may themselves delegate responsibility to other nodes, thus building a delegation chain. We will not present a description of how this works. The key idea in getting the configuration strategy to work is to show that after finitely many steps of the delegation process, it is safe for all nodes in the delegation chain to act simultaneously.

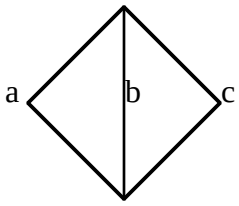

Figure 6.2

References

[A] C.J. Ash, Labelling systems and r.e. structures, Ann. Pure and Applied Logic *47*(1990), 99-119.

[C] S.B. Cooper, Degrees of Unsolvability, Ph.D. Dissertation, Leicester Univ., 1971.

[E] R. Epstein, Degrees of Unsolvability: Structure and Theory, Lecture Notes in Math. *759*, Springer-Verlag, Berlin, Heidelberg, New York, 1979.

[F] R.M. Friedberg, Two recursively enumerable sets of incomparable degrees of unsolvability, Proc. Nat. Acad. Sci. U.S.A. *43*(1957), 236-238.

[GS] M.J. Groszek and T.A. Slaman, Foundations of the Priority Method, I: Finite and infinite injury, (manuscript).

[H] P. Hinman, n-r.e. and n-REA sets, private communication.

[L1] A.H. Lachlan, Lower bounds for pairs of recursively enumerable degrees, Proc. London Math. Soc. *16*(1966), 537-569.

[L2] A.H. Lachlan, Bounding minimal pairs, J. Symbolic Logic *44*(1979), 626-642.

[LL1] S. Lempp and M. Lerman, Priority arguments using iterated trees of strategies, In: Recursion Theory Week, 1989, Lecture Notes in Mathematics #1482, Springer-Verlag, Berlin, Heidelberg, New York, 1990, pps. 277-296.

[LL2] Minimal pairs of r.e. degrees and iterated trees of strategies, In: Logical Methods, J. Crossley, J. Remmel, R. Shore, M. Sweedler eds., Birkhäuser, Boston, Basel, Berlin, to appear.

[LL3] S. Lempp and M. Lerman, The existential theory of the poset of r.e. degrees with a predicate for single jump reducibility, Jour. Symb. Logic, *57*(1992), 1120-1130.

[LL4] S. Lempp and M. Lerman, The decidability of existential theory of the poset of recursively enumerable degrees with jump relations, to appear.

[M] A. A. Muchnik, On the unsolvability of the problem of reducibility in the theory of algorithms, Dokl. Akad. Nauk SSSR, N.S. *108*(1956), 194-197 (Russian).

[Sa1] G.E. Sacks, Degrees of Unsolvability, Ann. Math. Studies no. 55, Princeton University Press, Princeton, N.J. 1963.

[Sl] T. A. Slaman, The density of infima in the recursively enumerable degrees, Ann. Pure and Applied Logic *52*(1991), 155-179.

[So] R. Soare, Recursively Enumerable Sets and Degrees, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, Heidelberg, New York, 1987.

[Y] C.E.M. Yates, A minimal pair of recursively enumerable degrees, J. Symbolic Logic *31*(1966), 159-168.