The p-T-degrees of the recursive sets: lattice embeddings, extensions of embeddings and the two-quantifier theory

Richard A. Shore*
Cornell University, Ithaca, NY, USA

Theodore A. Slaman**
University of Chicago, Chicago, IL, USA

Communicated by A.R. Meyer
Received May 1989
Revised April 1990

Abstract


Ambos-Spies (1984a) showed that the two basic nondistributive lattices can be embedded in $R_{p-T}$, the polynomial-time Turing degrees of the recursive sets. We introduce more general techniques to extend his results to show that every recursive lattice can be embedded in $R_{p-T}$. In addition to lattice-theoretic representation theorems, we use the scheme of priority style arguments coupled with "looking back" techniques presented in Shinoda and Slaman (1988, 1990). We also generalize the density type results of Ladner (1975) and many others to settle the full extension of the embedding problem for $R_{p-T}$. Combined with the logical analysis of sentences with one alternation of quantifiers (Shore 1978, Lerman 1983), these results suffice to decide the full $\forall \exists$-theory of $R_{p-T}$. They also give a strong nonhomogeneity result: the p-time degrees of the sets recursive in (and, if desired, p-time above) two distinct sets $A$ and $B$ are almost never isomorphic. The situation for the p-time many-one degrees is quite different. We decide the extension of the embedding problem (differently than for $R_{\leq_T}$) but not the $\forall \exists$-theory.

1. Introduction

A notion of reducibility $\leq_r$ between sets is specified by giving a set of procedures for computing one set from another. We say that a set $A$ is reducible to a set $B$, $A \leq_r B$, if

* Research partially supported by NSF Grants DMS-8601048 and DMS-8902797.
** Research partially supported by NSF Grant DMS-8601856 and PYI grant DMS-8451748.
one of the procedures applied to $B$ gives $A$. The most general notion of a computable reducibility is that of Turing, $\leq_T$. Here we say that $A \leq_T B$ if there is a Turing machine $\varphi$, which, when equipped with an oracle for $B$, computes $A$: $\varphi^B = A$. This is the most important measure of complexity on arbitrary noncomputable sets. When one is interested, instead, in analyzing the recursive sets in terms of relative complexity of computation or, indeed, simply in more practical resource-bounded measures of complexity, our accumulated experience seems to indicate that the correct (or, at least, the most important) relevant resource bound is that of polynomial time. Thus, the analog of Turing reducibility should be p-time reductions given by arbitrary Turing machines. This gives us the reducibility notion $\leq_{p-T}$ introduced by Cook [7]. The domain of discourse for considering any reducibility must also be carefully considered. Turing reductions are appropriate for the study of arbitrary sets. P-time Turing reductions seem most appropriate to the finer analysis of the recursive sets about which Turing reducibility, of course, has nothing to say. There are certainly many other important reducibilities in both the noncomputable and computable domains. We could mention one-one, many-one, truth-table (along with various bounded versions) and their polynomial-time analogs as familiar reducibilities stronger than Turing reducibility, as well as several interesting weaker ones in both settings. Of course, in the framework of the computable sets one must really consider other resource bounds (both in terms of time and space) as well. The types of sets considered can also be varied. On the one side, we can consider the recursively enumerable sets, the $\Delta^0_2$ sets (those recursive in $\emptyset'$), the arithmetic sets and more. Within the computable sets, we might mention those in NP, the polynomial hierarchy, PSPACE, exponential time, etc. In this paper we concentrate on p-time Turing reducibility on the recursive sets. Historical remarks and analogies to the noncomputable case will generally be confined to Turing reducibility. Other reducibilities and domains of discourse are mentioned to point out contrasts and areas for future work.

Once one has chosen a reducibility notion, the natural structure for investigation is that of the associated degrees, the equivalence classes determined by the reducibility relation ordered by the induced ordering. We denote these structures by letters hinting at the collection of sets considered and subscripted by an abbreviation for the reducibility. Thus, the structure of all the Turing degrees is denoted by $\mathcal{D}_T$ and that of the recursive sets under polynomial-time Turing reductions by $\mathcal{R}_{p-T}$. It is these structures in which we are interested. As usual, we denote the degrees which are their members by small roman boldface letters and a typical set in the degree by the corresponding capital italic lightface letter: $A \in a$.

The systematic study of the algebraic structure of $\mathcal{D}_T$ was begun by Kleene and Post [12] and has been a major area of research in recursion theory ever since. (We recommend [20] as a text on the general theory and [33] for $\mathcal{A}_T$, the Turing degrees of the recursively enumerable sets.) The corresponding work on $\mathcal{R}_{p-T}$ was initiated in [15, 16]. All these degree structures are upper semilattices (we will henceforth abbreviate upper semilattice as usl) and the algebraic investigations typically begin with questions that can be seen as embedding problems or their extensions. One begins with
just the partial order and asks, for example, if there are incomparable elements. This corresponds to embedding the diamond, i.e. the four element Boolean algebra, as a partial ordering (or, equivalently, as an usl). This question is answered affirmatively in both papers for their respective structures. Kleene and Post go on to show that, in fact, every countable partial order can be embedded into $\mathcal{D}_T$. For $R_{p-T}$ the corresponding result is an extension of Ladner’s work by Mehlhorn [22, 23] and independently by Breidbart [5].

After the embedding problem is settled, the next step in the typical structural analysis is the extension of embedding problem. In general, the problem is, given orderings $\mathcal{F} \subseteq \mathcal{Y}$, to decide if, for every embedding of $\mathcal{F}$ into our degree structure, there is an extension of this embedding to one of $\mathcal{Y}$. Here the typical first question is the density of the degrees: given a realization of the partial order $x_1 < x_2$ as $x_1 < x_2$, can it be extended to produce a $y$ between them? This question was left open in [12] but solved negatively in [34]: there are minimal Turing degrees. Ladner [15, 16], on the other hand, gives a positive solution for $R_{p-T}$: it is dense. Indeed, Mehlhorn [22, 23] shows that every countable partial order can be embedded in every interval of $R_{p-T}$.

Once beyond the question of density, the extension of embedding problem becomes intertwined with that of embedding lattices (with the lattice structure preserved). Neither degree structure is a lattice, but one can still ask (to start off) if there are two elements with an infimum or, more specifically, if there are two with inf 0 (also called a minimal pair). This corresponds to the following extension of embedding problem: given two incomparable degrees $x_1$ and $x_2$, can one always find a nonzero degree $y$ strictly less than both of them? Kleene and Post, as well as Ladner, show that minimal pairs exist and so give negative answers to this extension of embedding question. The problem of lattice embeddings in $\mathcal{D}_T$ was, after Spector’s construction of a minimal degree, subsumed under the more difficult problem of determining the initial segments of the structure. Work by many researchers over the years finally showed that any possible lattice, or even usl of various sizes (i.e. those with least element and the countable predecessor property), can be embedded as initial segments of $\mathcal{D}_T$ (see [19] for the finite ones, [14] for the countable ones and [1] for those of size $\aleph_1$.) As $R_{p-T}$ is dense, these results did not seem relevant. The important questions concerned simple lattice embeddings, perhaps preserving 0, but not as initial segments.

There were a number of papers that contributed to this next step of the analysis, particularly, in terms of developing structural techniques for constructing degrees with various properties. We should mention, in particular, [18, 6]. The next major contributions in terms of lattice embeddings are due to Ambos-Spies ([2, 4]; for the most complete account of these and related results see [3]). He proved that every countable distributive lattice can be embedded in $R_{p-T}$ as well as the two basic nondistributive ones: the pentagon ($N_5$) and 1-3-1 ($M_3$) of Fig. 1.

These results point to an important difference between $p-T$ reducibility and the $p-m$ (polynomial-time many–one) reducibility introduced by Karp [11]. Ambos-Spies
showed that $R_{p-m}$, in contrast to $R_{p-T}$, is a distributive usl. The general lattice embedding problem even for finite nondistributive lattices was, however, left open. We solve this problem by showing (Theorem 3.7) that every recursively presented lattice can be embedded in $R_{p-T}$. General considerations (as in [32]) based simply on the complexity of the ordering of $R_{p-T}$ and the existence of arbitrarily complicated finitely generated lattices show that one cannot hope to embed all, or even any sufficiently complicated, lattices in $R_{p-T}$ or, indeed, in any fixed countable structure. Such considerations show that the p-time degrees of sets recursive in (and also, if desired, p-time above) two distinct sets $A$ and $B$ are almost never isomorphic. Indeed, if these structures were isomorphic, our results would imply (Theorem 3.8) that the Turing degrees of $A$ and $B$ would be within a few jumps of each other.

Returning now to the general extension of embedding problem, we recall that Kleene and Post proved that any extension of a given embedding not requiring that new elements be put in below old ones can always be realized. We extend the previously known results on this problem due to Ladner, Ambos-Spies and others (mainly dense embedding ones combined with some cone-avoiding) to essentially show (Theorem 3.9) that any extension problem not violating the assumption that the given degrees form a lattice can always be realized.

All of these results can be viewed as attempts to decide fragments of the theory of the degree structure under investigation. Thus, the embedding of all countable (or even finite) partial orderings decides the one-quantifier theory, i.e. all sentences of the form $\exists x_1 \exists x_2 \ldots \exists x_n P(x_1, \ldots, x_n)$, where the quantifiers range over degrees, and $P$ is quantifier-free and contains only the predicate $\leq$ along with the usual logical connectives. (These sentences are called the $\exists$ sentences and the corresponding
fragment of the theory of a structure its $3$-theory.) Lattice embedding results can be seen as determining the $3$-theory of the structure in a richer language containing $\lor$ and $\land$. They are also an extension of embedding results in the language with just $\leq$. As such, they, along with all extensions of embedding results, can be seen as determining parts of the two-quantifier theory of the structure with $\leq$. Roughly speaking, if we are given the orderings $\{x_1, \ldots, x_n\} = \mathcal{X} \subset \mathcal{Y} = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$, the question is if, for every $x_1, \ldots, x_n$ satisfying the orderings prescribed by $\mathcal{X}$, we can always find $y_1, \ldots, y_m$ ordered, as are the elements of $\mathcal{Y}$. Such questions are of the form $\forall x_1, \ldots, x_n \exists y_1, \ldots, y_m (D_x(x_1, \ldots, x_n) \rightarrow D_y(x_1, \ldots, x_n, y_1, \ldots, y_m))$, where $D_x$ and $D_y$ are the quantifier-free complete diagrams of the respective orderings. Now, in general, not all $\forall \exists$ sentences, i.e. those with only one alternation of quantifiers or, more explicitly, ones of the form $\forall x_1, \ldots, \forall x_n \exists y_1, \ldots, \exists y_m P(x_1, \ldots, x_n, y_1, \ldots, y_m)$ with $P$ quantifier-free, are equivalent to an extension of embedding problems. In the case of $\exists^1$, however, the initial segment results of Lerman [19] combined with the extension of embedding results of Kleene and Post solve the extension of embedding problem in such a way as to actually decide the full $\forall \exists$-theory of the structure. Although the theories are quite different, it also turns out that in the case of $\mathcal{R}_{p-T}$ our lattice embedding and extension results also solve the extension of embedding problem in a way that allows us to decide the full $\forall \exists$-theory (Theorem 3.11). This answers a question of Shinoda and Slaman [29, 30]. We give the results of the logical analysis of the $\forall \exists$ sentences that reduces their decidability to extension of embedding questions for $\mathcal{R}_{p-T}$ in Section 2. It is worth pointing out, however, that the analysis does not give such a reduction for $\mathcal{R}_{p-m}$ because of structural differences between it and $\mathcal{R}_{p-T}$. We discuss these problems in Section 4.

In the analysis of $\mathcal{D}_T$ the initial segment results that play a key role in the decidability of the $\forall \exists$-theory also prove the undecidability of the full theory. The point there is that, as initial segments form a definable class of substructures, the undecidable theory of (even distributive) lattices can be interpreted in $\mathcal{D}_T$ to get its undecidability [13]. Indeed, a careful analysis of the interpretation of the theory of graphs into that of lattices shows that the three quantifier or $\forall \exists \forall$-theory of $\mathcal{D}_T$ is also undecidable [20, Ch. VII, Section 4.6]. Thus, the precise level of undecidable fragments has been determined. Once one knows that all finite lattices can be embedded as initial segments of $\mathcal{D}_T (\leq 0')$, the Turing degrees below that of the halting problem [20, Ch. XII], precisely the same analysis gives the undecidability of its $\forall \exists \forall$-theory. The decidability of its $\forall \exists$-theory requires a new extension of embedding argument but then proceeds similarly [21]. The situation for $\mathcal{R}_{p-T}$ is quite different. It is known to be undecidable [29, 30]; indeed, like $\mathcal{D}_T$ and $\mathcal{D}_T (\leq 0')$, its theory is as complicated as possible, true second-order arithmetic for $\mathcal{D}_T$ and true first-order arithmetic for $\mathcal{D}_T (\leq 0')$ and $\mathcal{R}_{p-T}$. The coding used in [29, 30] is, however, very complicated (much more like the one used for $\mathcal{R}_T$ than $\mathcal{D}_T$) and the gap between the $\forall \exists$-theory that we here show decidable and the fragment needed there to give undecidability is large.

We describe the logical analysis of $\forall \exists$ sentences and the lattice representations we need in Section 2. In Section 3 we give first an overview of the procedures used in the
past for constructions of p-time degrees and then an intuitive sketch of our construction methods. We then give the actual construction of an embedding of a recursive lattice in $R_{p-T}$ and of the extension of embedding theorem needed to complete the decision procedure for its $\forall \exists$ sentences. As a corollary to the lattice embedding results, we show that the relativized versions of $R_{p-T}$ are almost never isomorphic. Section 4 discusses the analogous questions for $R_{p,m}$ and other open questions.

2. Two-quantifier sentences and lattice representations

There are three main components of our proofs in terms of techniques of analysis and construction. The first is the logical analysis of $\forall \exists$ sentences that reduces their decision problem to a form that can be solved, at least in some cases, by appropriate lattice embedding and extension results. The second is the lattice-theoretic approach needed to represent all recursive lattices in a form suitable for coding into the p-time Turing degrees of recursive sets. Finally, the most important considerations concern the actual construction of the desired lattices and extensions in $R_{p-T}$.

The first step is common to all such analyses of usl’s and follows the procedure used for the Turing degrees used in [31] and described in detail in [20, Ch. VII, Section 4]. As $0$, the least element of the ordering, is definable within any $\forall \exists$ sentence without increasing its quantifier complexity, there is no essential difference at this level between the language of partial orders with just $\leq$ and that with a constant for $0$ as well. It is, however, more convenient to work in the language with $0$. The analysis given in [20] applied to this language reduces the problem of deciding all $\forall \exists$ sentences in a fixed usl with least element $0$ to deciding the truth of the ones of the form

$$\Gamma(x_1, \ldots, x_n, y_1, \ldots, y_m) = \forall x_1, \ldots, x_n \exists y_1, \ldots, y_m (\Theta(x_1, \ldots, x_n) \rightarrow \forall i<k \Psi_i(x_1, \ldots, x_n, y_1, \ldots, y_m)),$$

where $\Theta$ is a complete diagram in the variables $x_1, \ldots, x_n$ in which every pair of elements has a least upper bound and each $\Psi_i$ is a complete diagram in the variables $x_1, \ldots, x_n, y_1, \ldots, y_m$. (A complete diagram in the language of partial orderings for the variables $z_1, \ldots, z_m$ is just a list including for each pair of variables $z_i$ and $z_j$ either $z_i \leq z_j$ or $z_i \not\leq z_j$, but not both. Without loss of generality, we may assume that all the variables represent distinct elements, i.e., for $i \neq j$, we must have $z_i \leq z_j$ or $z_j \not\leq z_i$. As we are working in the language with $0$, we also include the facts that $0 \leq z_i$ and $z_i \not\leq 0$ for every $z_i$ in every complete diagram.)

Consider now a sentence $\Gamma(x_1, \ldots, x_n, y_1, \ldots, y_m)$ as given by this analysis. As every finite usl with 0 is a lattice (the infimum of $x$ and $y$ is the supremum of the finitely many elements below both of them), the diagram $\Theta$ actually specifies a finite lattice $\mathcal{L}$. Once we know (Theorem 3.7) that this lattice is embeddable (as a lattice with 0 preserved) in $R_{p-T}$ (with a realization, say, by p-T degrees $0, a_j, \ldots, a_n$), it is clear that, if $\Psi_i$ is
inconsistent with the lattice structure of $\mathcal{L}$, then there is no way to extend the embedding to degrees $b_1, \ldots, b_m$ which will make $\Psi_i(a_1, \ldots, a_n, b_1, \ldots, b_m)$ true. (We say that $\Psi$ is inconsistent with the lattice structure of $\mathcal{L}$ if there is some ordering fact in $\Psi$ which contradicts some ordering, join or infimum fact about $\mathcal{L}$. A typical example is that $\Psi$ requires some $y$ to be less than both $x_1$ and $x_2$ but not less than $x_3$ while, in $P'$, $x_3$ is the infimum of $x_1$ and $x_2$.) Thus, if every $\Psi_i$ is inconsistent with the lattice structure of $\mathcal{L}$, the original sentence $\Gamma$ is false in $\mathbb{R}_{p-T}$. On the other hand, if any $\Psi_i$ is consistent with $\mathcal{L}$, our extension of embedding result (Theorem 3.9) will say precisely that for every choice of degrees $a_1, \ldots, a_n$ making $\Theta$ true there are $b_1, \ldots, b_m$ such that $\Psi_i(a_1, \ldots, a_n, b_1, \ldots, b_m)$ is true in $\mathbb{R}_{p-T}$. In this case the original sentence $\Gamma$ is true. Thus, we will have decided all sentences of this form and, so, all $\forall \exists$ sentences in $\mathbb{R}_{p-T}$.

The lattice-theoretic representations that we use for our coding scheme are also familiar in essence from initial segment results and simple lattice embeddings in $\mathbb{R}_T$ as in [20] and [32], respectively. They are equivalent to the dual of the standard representation of lattices in terms of equivalence relations (as in any text on lattices, such as [9]). As in [32], we do not need the homogeneity property required in initial segment results as we are simply embedding the lattices in $\mathbb{R}_{p-T}$. We will, however, have to take some precautions not needed there to keep our alphabet finite. We begin with the basic definitions.

**Definition 2.1 (Lattice representations).** Let $\mathcal{L}$ be a lattice (with least element 0) with universe $L$ and relations $\leq$, $\vee$ and $\wedge$.

If $\alpha$ and $\beta$ are functions from $L$ into $\mathbb{N}$ and $i \in L$, we say that $\alpha$ is congruent to $\beta$ modulo $i$, $\alpha \equiv_i \beta$ if $\alpha(i) = \beta(i)$.

A set $\Theta$ of functions $\alpha: L \rightarrow \mathbb{N}$ is an usl representation of $\mathcal{L}$ if, for all $\alpha, \beta \in \Theta$ and $i, j, k \in L$, the following conditions are met:

1. **Zero:** $\alpha \equiv_0 \beta$.
2. **Ordering:** $i \leq j$ & $\alpha \equiv_j \beta$ $\Rightarrow$ $\alpha \equiv_i \beta$.
3. **Nonordering:** $i \not\leq j \Rightarrow \exists \alpha, \beta \in \Theta$ ($\alpha \equiv_j \beta$ & $\alpha \not\equiv_i \beta$).
4. **Join:** $(i \vee j - k)$ & $\alpha \equiv_i \beta$ & $\alpha \equiv_j \beta$ $\Rightarrow$ $\alpha \equiv_k \beta$.

Without loss of generality, we may assume that $\alpha(0) = 0$ for every $\alpha \in \Theta$ and for convenience that there is an $\alpha_0 \in \Theta$ which is identically 0. For notational convenience we also denote this element of any representation by 0.

We say that an usl representation $\Theta$ is a lattice representation if, in addition, the following condition is met:

1. **Meet:** For every $\alpha, \beta \in \Theta$ and every $i, j$ and $k$ in $\mathcal{L}$, with $i \wedge j = k$ and $\alpha \equiv_k \beta$, there are $\gamma_1, \gamma_2$ and $\gamma_3$ in $\Theta$ such that $\alpha \equiv_i \gamma_1 \equiv_j \gamma_2 \equiv_i \gamma_3 \equiv_j \beta$.

Our plan is to use these lattice representations to carry much of the burden of our embedding procedure. Suppose that we have a finite lattice $\mathcal{L}$ which has a finite lattice representation $\Theta$. Let $\Sigma \subset \mathbb{N}$ be the (finite set which consists of the) range of elements from $\Theta$. Our construction will build a recursive function $g: \mathbb{N} \rightarrow \Theta$ from which we will
define for each \( i \in L \) certain sparse subsets \( G_i \) of \( \Sigma^* \), the set of finite strings from \( \Sigma \). The embedding from \( \mathcal{L} \) into \( \mathcal{R}_{p-T} \) will then be given by sending \( i \) to the degree of \( G_i \).

**Definition 2.2 (Projections).** If \( g \) is a map from \( \mathbb{N} \) into \( \Theta \), we define the projections \( G_i \) of \( g \) for \( i \in L \) by setting \( G_i = \{0^n \cdot (g(n))((i)) \mid n \in \mathbb{N}\} \). We denote the p-time degree of \( G_i \) by \( g_i \). The map sending \( i \in L \) to \( g_i \) is the embedding of \( \mathcal{L} \) into \( \mathcal{R}_{p-T} \) induced by \( g \) (and \( \Theta \)).

It is now fairly easy to see that the embedding induced by any map \( g : \mathbb{N} \to \Theta \) preserves zero, order and join.

**Lemma 2.3.** For any \( g : \mathbb{N} \to \Theta, G_0 \equiv_{p-T} 0 \).

**Proof.** By definition \( G_0 = \{0^n+1 \mid n \in \mathbb{N}\} \).

**Lemma 2.4.** For any \( g : \mathbb{N} \to \Theta, i \leq j \Rightarrow g_i \leq g_j \).

**Proof.** Consider any \( \tau \in \Sigma^* : \tau \in G_i \Leftrightarrow \tau \) is of the form \( 0^n \cdot x \) and \( (g(n))(i) = x \). To decide if the latter condition holds, we simply find any \( x \) in the finite set \( \Theta \) with \( 0^n \cdot x(j) \in G_j \).

The definition of the projections guarantees that there is such an \( x \) and the ordering condition (2.1.1) in the definition of a lattice representation guarantees that there is only one value for \( x(i) \) over all such \( x \)'s. As this value must then be \( (g(n))(i), \tau \in G_i \) iff \( x(i) = x \).

**Lemma 2.5.** For any \( g : \mathbb{N} \to \Theta, i \vee j = k \Rightarrow g_i \vee g_j = g_k \).

**Proof.** To see if \( \tau = 0^n \cdot x \) is in \( G_k \), find any \( x \in \Theta \) such that \( 0^n \cdot x(i) \in G_i \) and \( 0^n \cdot x(j) \in G_j \).

The join requirement (2.1.3) of the definition then tells us that \( \tau \) is in \( G_k \) iff \( x(k) = x \).

Thus, if we build any \( g : \mathbb{N} \to \Theta \) and use these projections, we will define an embedding of \( \mathcal{L} \) into \( \mathcal{R}_{p-T} \) which preserves zero, order and join. The task of our construction will then be to guarantee that we preserve \( \preceq \) and \( \wedge \).

In addition, we must be much more careful with the construction of the representation than in the Turing degree case. As the representation is used in the decoding, it is not sufficient to build a recursive representation for a recursive lattice (as in [32]) as this will introduce non-p-time reductions into procedures that should be simple table look-up. Thus, we really need finite representations. Our salvation here is a quite difficult theorem of lattice theory.

**Theorem 2.6 (Pudlak and Tuma [25]).** Every finite lattice has a finite representation as a lattice of equivalence relations (and, so, a finite lattice representation in our sense).

If we wish to embed infinite recursive lattices, the situation is a bit more complicated as they cannot have finite representations. We can, however, use the representa-
tions guaranteed by this theorem to build a sequence of finite representations for each of a recursive sequence of finite sub-usl's of the given lattice. Let \( L \) be a given recursive lattice with domain \( \mathbb{N} \). Let \( L_n \) be a uniformly recursive sequence of finite sub-usl's of \( L \) whose union is all of \( L \). (Note that any finite subset of a lattice generates a finite usl. The set of all finite joins of elements from any subset \( F \) of the lattice is itself closed under joins. The sublattice generated by a finite set, on the other hand, need not be finite.) As every finite usl is a lattice, we can view each \( L_n \) as a finite lattice (albeit not necessarily a sublattice of \( L \) as the infima may be different). By Theorem 2.2 there are finite lattice representations \( \Theta_n \) for each \( L_n \). As they are finite and being a representation for \( L_n \) is uniformly recursive, we can take the \( \Theta_n \) to be a recursive sequence.

**Theorem 2.7.** If \( L \) is a recursive lattice, there is a recursive sequence of finite sub-usl's \( L_n \) of \( L \) with union \( L \) and a recursive sequence of finite lattice representations \( \Theta_n \) for \( L_n \).

When we use this sequence in the construction, we spread its elements far enough apart so that the table look-up procedures needed to decode at any particular string become polynomial in the length of the string. We also need to show that it suffices to eventually have the correct infimum for each pair of elements to satisfy the embedding requirements in the construction.

### 3. The constructions

Finally, we come to the heart of our construction, actually building the recursive sets so as to meet the requirements of the lattice embedding. The coding procedure implicit in the use of lattice representations guarantees that zero, ordering and join are automatically preserved. Our concerns are, therefore, twofold. We must first meet the requirements corresponding to diagonalizations to guarantee that if, in \( L \), \( i < j \), then the degrees \( g_i \) and \( g_j \), realizing them are such that \( g_i < g_j \). Secondly, we must set up and meet requirements that guarantee that if, in \( L \), \( i \wedge j = k \), then the corresponding degrees have the same infimum property: \( g_i \wedge g_j = g_k \).

Typically, embedding and extension of embedding results have been proven by delayed diagonalization arguments introduced by Ladner \([15,16]\) and used by Mehlhorn, Breidbart and others for such purposes. Structural versions of these methods were later introduced by Landweber et al. \([18]\). These were further developed by Chew and Machtley \([6]\), Schöning \([27,28]\), Ambos-Spies \([2-4]\) and others. Such arguments suffice to embed all distributive lattices in \( \mathbb{R}_{\text{p-T}} \) (indeed even densely) as in Ambos-Spies \([4]\). They cannot, however, be applied to get our lattice embedding results. These techniques were designed for, and supply constructions that, work below any non-zero p-T degree. Thus, they are suitable for positive extension of embedding results and we, in fact, use them for our results of this type. On the other
hand, Ambos-Spies [3] has shown that there are recursive p-T degrees below which the p-T degrees are distributive. Thus, we cannot hope to exploit these techniques to embed nondistributive lattices. In addition, the structural versions are closely tied with codings exploiting set-theoretic containment, union and intersection. Thus, they are unsuited to our purposes even if freed of the aspects forcing them below an arbitrary degree. We must formulate and tackle our requirements directly.

Ambos-Spies introduces a new technique for preserving infima in [3, Ch. III, Section 10] to embed the two basic five-element nondistributive lattices in $\mathbb{R}_{p-T}$. His procedure, however, is not sufficiently general to handle arbitrary (even finite) lattices. We must, therefore, use a more complicated way of presenting and attacking even a single requirement that is tied to our representation of lattices. An important and interesting aspect of the construction is the way the requirements are put together. Our general plan of attack on constructions of p-T degrees extends that of Ambos-Spies [3] and follows that of Shinoda and Slaman [29, 30] who borrow heavily from both the language and insight developed for forcing in set theory and recursion theory and the style and organization of priority arguments invented for constructions of recursively enumerable sets. We first describe our construction briefly in the abstract and then give the full details. As the form of the requirements here is much simpler than that used in [29, 30], we hope that this may make the methods somewhat more accessible. It should have many further applications.

The standard simple and delayed diagonalization arguments have the feature that we know that we can always diagonalize to satisfy a requirement. We may have to continue some operation such as copying in a given non-p-time set or a constant string of zeros for some undetermined number of stages. In the end, however, we can wait until the requirement has been met. We can also recognize that it has been met when we do so. Our requirements are at one-quantifier level higher. We are searching for some extension of the set constructed so far and recognizable computations which will guarantee meeting the requirement as by a diagonalization of some sort. The problem is that there may be no such extension or computations. In this case, we must argue, after the construction is over, that we somehow managed to satisfy the requirement in some other way. It is this uncertainty as to how we will ultimately satisfy the requirements that forces us to a priority style construction. The situation is analogous to the hierarchy of priority arguments for r.e. sets beginning with wait-and-see arguments and then progressing to finite, infinite and monstrous injury. These injury arguments are also classified as $0'$, $0''$ and $0'''$ constructions in accordance with the difficulty of determining how the requirements are satisfied. In this scheme the typical (delayed) diagonalization constructions for $\mathbb{R}_{p-T}$ might well be called recursive. In our construction, as in the one for embedding recursive lattices in the Turing degrees below $0'$ (viewed as a recursive full approximation construction with priorities), it takes a $0'$ oracle to determine how the requirements are satisfied. It would, therefore, correspond to a finite injury argument. (The requirements considered in [29, 30] correspond to ones that in the context of r.e. sets demand $0'''$ constructions. The strategies there are, thus, more complicated than the ones needed here.)
Although we would characterize our construction as a finite injury one in the classical recursion-theoretic sense, its distinguishing feature from the viewpoint of complexity-theoretic arguments is the “look ahead” procedure employed to satisfy the infimum requirements of the construction. (A general description of the style of our constructions is given in the following paragraphs. More specifically, the “look ahead” technique is implemented in the second paragraph of the description of substage $u$ of the construction. Its effects are analyzed and exploited in the proof of Lemma 3.6.)

A rough idea is to bound a search for a diagonalization type witness in such a way that certain calculations can be made based on the fact that no such witnesses were found below the prescribed bound. This type of procedure does not seem to appear in other simpler constructions that might also be called finite-injury constructions such as those of Homer and Maass [10]. It seems to have first appeared in [3] and then, in a more complicated setting, in [29, 30]. A formal analysis of the complexity of some of these arguments in complexity theory can be found in [26].

In short, our construction proceeds as follows. At a stage $s$, we have determined some initial segment $p^s$ of $g$, the function that we are building, and have a list $R_{s_1}, \ldots, R_{s_n}$ of requirements which we have not yet met in a recognizable way. We now perform some search for an extension $p'$ of $p^s$ and a recognizable witness to meeting the last (and, so, the lowest priority) requirement $R_{s_n}$. If we find such an extension and witness, then we try to adopt this extension as an initial segment of $g$. If no requirement of higher priority finds an extension suitable for its needs, we in fact adopt the one found for $R_{s_n}$. We then implement searches for each requirement in turn with increasing priority. If a higher priority requirement does act, it cancels the witness and restrictions imposed by lower priority ones. The key to making everything needed polynomial-time is combining a version of the “looking back” technique in the proof with the right “look ahead” technique in the construction. For the sake of $R_{s_n}$ we search, roughly speaking, over all strings $\sigma$ such that $|\sigma|$ is less than or equal to the number of steps needed to compute the entire course of the construction up to the beginning of stage $s$. (Although this seems to invoke the recursion theorem, we do not actually need to do so. Attaching a counter to the construction and appropriate polynomial-time manipulations of $|\sigma|$ will suffice. Regan [26] also comments on this point.) For the sake of $R_{s_{n-1}}$ we search over all strings whose length is at most the number of steps needed to compute the result of the search for $R_{s_n}$. We continue on in this way through all the requirements. At the end of these searches we adopt the extension found satisfying the requirement of highest possible priority.

If we never find a recognizable witness to the satisfaction of some requirement $R$ (corresponding to an infimum preserving condition such as $g_i \land g_j = g_k$), we will have to prove that some set computed in polynomial time from both $G_i$ and $G_j$ (the sets we construct of the desired degrees) is also p-time computable from $G_i$. The idea is to arrange the construction so that (after all requirements of higher priority have settled down) for any (sufficiently long) string $\sigma$, the search which considered $\sigma$ took place at a stage $s$ such that in (polynomially in) $|\sigma|$ steps we can calculate the entire construction up to the point within stage $s - 1$ at which we began the search for a witness to
$R$ over a set of strings including $\sigma$. Our assumption that the higher priority requirements have settled down and that we do not find a witness for $R$ itself means that in (polynomially in $|\sigma|$) steps we have also calculated the true outcome of stage $s-1$. Thus, we know the segment $p^i$ that we try to extend when we search over a set of strings containing $\sigma$. The priority arrangement guarantees that no action for requirements of lower priority than $R$ at stage $s$ can prevent us from taking an extension which supplies a witness to a recognizable win, if one exists. The assumption that none exists allows us to correctly compute the desired function at $\sigma$ by computing relative to a trivial extension of $p$.

Our goal now is to eventually prove our main embedding result.

**Theorem 3.7.** Every recursive lattice $\mathcal{L}$ can be embedded in the polynomial-time Turing degrees of the recursive sets.

To eliminate some of the coding problems, we first describe the case of embedding a finite lattice $\mathcal{L}$ with universe $L$ and finite representation $\Theta$ with range $\Sigma$, a finite set of natural numbers. We begin the technical description of the construction with our approximation schemes and precise statement of the requirements. As explained in the previous section, we build a function $g: \mathbb{N} \rightarrow \Theta$ and define our embedding by sending $i \in L$ to $G_i \subseteq \Sigma^*$. After we finish the finite lattice version of the construction, we will explain the changes needed to deal with arbitrary recursive lattices.

### 3.1. Approximations and forcing

We approximate the function $g$ that we are building by finite initial segments $p$. In the usual way we call the domain of $p$ its length and denote it by $lth(p)$. We call the set $\mathcal{P}$ of all such $p$ our notion of forcing. We interpret the projection functions in the obvious way: $P_i \equiv \{0^n \wedge (p(n))(i) \mid n < lth(p)\}$. We say that $p$ forces, written as $p \models$, some basic sentence of the form $\phi_a^a \equiv G_i(\sigma) \land q \neq G_j(\sigma) \lor \phi_a^a \equiv G_j(\sigma) \land q \neq G_i(\sigma)$ if $\phi_a^a \equiv G_i(\sigma) \land q \neq G_j(\sigma)$ or $\phi_a^a \equiv G_j(\sigma) \land q \neq G_i(\sigma)$, respectively. In these latter assertions we intend to include the stipulation that no questions are asked of the oracle in the relevant computations about strings of length greater than the length of $p$. Thus, if $g \models p$ and $p$ forces one such sentence, then the corresponding one will be true of $G_i$ and $G_j$.

### 3.2. The requirements

We let $\varphi_e$ be a list of the polynomial-time Turing machines computing characteristic functions from every oracle. For the sake of definiteness we assume that on an input of length $n$ $\varphi_e$ runs at most $n^e + e$ steps. Our requirements are of two types:

- $D_{e,i,j}$: if $i \leq j$, then $\phi_{e,j} \neq G_i$.
- $M_{a,b,i,j}$: if $i \leq j = k$ and $\varphi_{a,i} = \varphi_{b,j} = C$, then $C$ is $p$-T reducible to $G_k$. 


The p-T-degrees of the recursive sets

We try to satisfy a requirement $D_{e,i,j}$ by diagonalizing, i.e. we try to find a $\sigma$ and an approximation $p$ to $g$ such that $p \vdash \varphi^G_i(\sigma) \neq G_i(\sigma)$. The nonordering requirement of the definition of a lattice representation (2.1.2) guarantees that we can always find such diagonalization witnesses. The requirements $M_{a,b,i,j}$ are more complicated. We try to actively meet them by attempting to find a $c$ and an approximation $p$ to $g$ such that $p \vdash \varphi^G_i(\sigma) \neq \varphi^G_i(\sigma)$. If we are never able to find such an approximation $p$ and string $\sigma$, we must argue that the requirement is satisfied by showing that we can compute $C$ from $G_k$ in polynomial time. Here the meet condition (2.1.4) plays a crucial role.

We order all the requirements in a single list $R_n$, $n \in \mathbb{N}$.

3.3. The construction

At the beginning of each stage $s$ of the construction we have an approximation $p^s$ to $g$, a list of requirements $R_{m_0}, R_{m_1}, \ldots, R_{m_{m(s)}}$ which have not yet been satisfied, with $n_{m(s)} = s$ and all other indices being $< s$ (once put on the list at the beginning of stage $s$, $R_s$ stays on the list until satisfied at which point it is removed); and a counter $c(s)$ set to some number in unary notation. As the construction proceeds, the counter is incremented by one at every step of the machine carrying out the construction other than those needed to increment the counter. (We picture the machine carrying out the construction as having an extra tape with two heads to do the counting. One head is always at the last symbol on the tape and writes a 1 and moves one square to the right whenever any other action is taken by the machine except at the end of a stage or substage of the construction. At the end of a stage it writes a 0 and then a 1. At the end of a substage which is not the end of a stage, it writes 001. The other head replaces the previous 0 or 00 with 1 or 11, respectively, when the first head writes the new zeros. It is also used to run over the string of 1's up to the first 0 or 00 as needed for certain counting comparisons in the construction.) At the beginning of stage 0 we have $c(0) = 0$ and we set $p^0 = 0$. The list of requirements is just $R_0$. Nothing actually happens at stage 0. We now consider stages $s > 0$.

Stage $s$: This stage consists of $m(s) + 1$ substages corresponding to the requirements on our list in reverse order. Thus, at substage 0 we consider the requirement $R_{m(s)}$, and at substage $u$ we consider $R_{m(u)-u}$. If at substage $u$ we find witnesses to satisfy the requirement we are considering, we define an extension $p^{s+u}$ of $p^s$. We let $n$ be the length of $p^s$ and $c(s,u)$, the value of the counter at the beginning of the $u$th substage. At the end of the last substage we finish the stage by defining $p^{s+1}$ to be $p^{s+u}$ for the largest $u$ for which such an extension is defined. This then satisfies $R_{m(u)-u}$, the highest priority requirement we could satisfy at this stage, and we remove it from our list. If no $p^{s+u}$ is defined, we simply set $p^{s+1} = p^s$ to conclude stage $s$.

Substage $u$: If $R_{m(u)-u}$ is $D_{e,i,j}$, we choose $\alpha, \beta \in \Theta$ as given by the nonordering property (2.1.2) of the definition of a lattice representation, i.e. $\alpha \equiv \beta$ but $\alpha \not\equiv \beta$. We define extensions $p^\alpha$ and $p^\beta$ of $p^s$ by setting $p^\alpha(n) = \alpha$, $p^\beta(n) = \beta$ and letting both be 0 for all numbers in $(n, (n + 1)s + e]$. Note that as $\alpha \equiv \beta$, $P^\alpha = P^\beta$. On the other hand, as $\alpha$ and $\beta$ are not congruent modulo $i$, $0^n \alpha(i) \in P^\alpha$ but $0^n \beta(i) \not\in P^\beta$. Now if $\sigma = 0^n \alpha(i)$,
then both $p^a$ and $p^b$ are long enough to force a value (indeed, by the above remark, the same value) for $\varphi^{G_i}(\sigma)$. As they differ at $\sigma$, one of $P^1_i$ and $P^2_i$ must differ from this value at $\sigma$. Let $p^{i,*}$ correspond to the one which so differs. Thus, $p^{i,*} \vdash \varphi^{G_i}(\sigma) \neq G_i(\sigma)$, as desired.

If $R_{m(s)-u}$ is $M_{a,b,i,j}$, we search over all $\sigma \in \Sigma^*$ of length at most $c(s,u)$ and over all $p \in \mathcal{P}$ of length at most $c(s,u) + d$, where $d$ is the maximum of $a$ and $b$, for a $p \supset p^i$ and a $\sigma$ such that $p \vdash \varphi^{G_i}(\sigma) \neq \varphi^{G_j}(\sigma)$. If we find such a $p$ and $\sigma$, we set $p^{i,*}$ equal to the first such $p$ that we found. Otherwise, $p^{i,u}$ is not defined. (We will see later why this eventually gives the desired conclusion about computing from $G_k$.)

We must now argue that all the requirements are met. First, it is clear that if we ever act to satisfy a requirement of either type and remove it from the list, then that requirement is met. The diagonalization requirements are now handled easily.

**Lemma 3.4.** The requirements $D_{e,i,j}$ are all met.

**Proof.** Suppose $D_{e,i,j}$ is $R_n$. Let $s > n$ be sufficiently large so that all requirements $R_m$ for $m < n$ which are ever removed from the list have already been removed. If we have already removed $R_n$ from our list, it has been satisfied as required. If not, then at the substage $u$ of stage $s$ at which we consider $R_n$ we define an extension $p^{i,u}$ of $p^i$ such that $p^{i,u} \vdash \varphi^{G_i}(\sigma) \neq G_i(\sigma)$ for some $\sigma$. By our assumption, no requirement of priority higher than $R_n$ can be satisfied at stage $s$. Thus, we set $p^{i,n+1} = p^{i,u}$ at the end of stage $s$ and, so, satisfy the requirement (and remove it from our list).

The argument for the requirements $M_{a,b,i,j}$ is a bit more complicated. We first need a lemma that explains how we exploit the meet requirement (2.1.4) in the definition of a lattice representation.

**Lemma 3.5.** If there are $q$ and $s$ in $\mathcal{P}$ of length $t = nt + d$ ($d = \max \{a, b\}$) both extending a $p'$ of length $m$ with $Q_k = S_k$ such that $q \vdash \varphi^{G_i}(\sigma) = x$ and $s \vdash \varphi^{G_j}(\sigma) = y$ for some $\sigma$ of length $n$ and some $x \neq y$, then there is a $p \in \mathcal{P}$ (of length $t$) extending $p'$ with $P_k = Q_k = S_k$ such that $p \vdash \varphi^{G_i}(\sigma) \neq \varphi^{G_j}(\sigma)$.

**Proof.** Note first that, as neither $\varphi_a$ nor $\varphi_b$ can ask any oracle question of length greater than $t$ on input $\sigma$, any $p$ of length $t$ forces some value for $\varphi^{G_z}(\sigma)$ for $z = i, j$ and $c = a, b$. The meet requirement (2.1.4) in the definition of a representation then guarantees, for each $v$ with $m \leq v < t$, the existence of elements $\gamma_{v,1}, \gamma_{v,2}$ and $\gamma_{v,3}$ such that $q(v) = \gamma_{v,1}, \gamma_{v,2}$ and $\gamma_{v,3} = s(v)$. Define $p^w$ by setting $p^w(v) = \gamma_{v,w}$ for $m \leq v < t$ and $w = 1, 2, 3$. Our choice of the $\gamma_{v,w}$ guarantees that $Q_i = P^1_i, P^1_j = P^2_j, P^2_i = P^3_i$ and $P^3_j = S_j$. In particular, any value for a computation from $G_i$ forced by $q$ is forced by $p^1$; any value for a computation from $G_j$ forced by $p^1$ is also forced by $p^2$; etc. We must, therefore, have $p^1 \vdash \varphi^{G_i}(\sigma) = x$ and $p^3 \vdash \varphi^{G_j}(\sigma) = y$. If no $p^w$ forced $\varphi_a(\sigma) \neq \varphi_b(\sigma)$, we would have $p^1 \vdash x = \varphi^{G_i}(\sigma) = \varphi^{G_j}(\sigma), p^2 \vdash \varphi^{G_j}(\sigma) = \varphi^{G_i}(\sigma) = x; p^3 \vdash x = \varphi^{G_i}(\sigma) = \varphi^{G_j}(\sigma)$ and, so, $s \vdash x = \varphi^{G_j}(\sigma)$ for a contradiction.
Lemma 3.6. The requirements $M_{a,b,i,j}$ are all met.

Proof. Suppose $M_{a,b,i,j}$ is $R_m$. Assume that $\varphi_a^{G_i} = \varphi_b^{G_j} = C$ (and, so, $R_m$ is never taken off the list of requirements after it is put on at stage $m$). Let $s_0 > m$ be a stage by which every requirement of priority higher than $R_m$ that is ever taken off our list has already been removed. Consider any $\sigma$ of length $n > c(s_0)$. We wish to compute $C(\sigma)$ (uniformly in $\sigma$) from $G_k$ (in polynomially) in $n$ steps. We run our construction until the counter has value $n$. This takes only a constant times $n$ steps. Let $u$ be the first substage, not yet started, of the last stage $s > s_0$ of the construction that has been started when the counter reaches $n$. For convenience we divide the argument into two cases depending on whether we have begun our consideration of requirement $R_m$ at stage $s$ when the counter reaches $n$.

If we have not yet begun to consider $R_m$, then our search at stage $s$ for the sake of satisfying $R_m$ will include $\sigma$ in its scope. As we have already begun stage $s$, we have correctly computed $p^\sigma$. We know that at the appropriate substage $v$ of stage $s$ we will search for a $p \geq p^\sigma$ which forces $\varphi_a^{G_i}(\sigma) \neq \varphi_b^{G_j}(\sigma)$. Were we to find one, we would define $p^{v+1}$. As no requirement of priority higher than $m$ is satisfied at $s$ by our choice of $s_0$, we would set $p^{v+1} = p^{v+1}$ if it were defined and so satisfy $R_m$. As this would contradict our assumption that $\varphi_a^{G_i} = \varphi_b^{G_j}$, we know that there is no such $p \geq p^\sigma$. By Lemma 3.5, every $q, q' \geq p^\sigma$ long enough to force any value for $\varphi_a^{G_i}(\sigma)$ and $\varphi_b^{G_j}(\sigma)$ such that $Q_k Q_k \subset G_k$ makes them converge to the same value as the true value of $C(\sigma)$. Thus, to correctly compute $C(\sigma)$, all we need to do is calculate $\varphi_a^{G_i}(\sigma)$, where $p$ extends $p^\sigma$ and, for $\text{lth}(p^\sigma) \leq m < n^a + a$, $p(m) = \alpha_m$ for $\alpha_m$, the least element of $\Theta$, such that $0^b \alpha_m(k) \in G_k$. As this is obviously polynomial in $n$ with an oracle for $G_k$, we have successfully computed $C(\sigma)$ from $G_k$, as required.

Finally, suppose that we have at least begun the substage of stage $s$ at which we consider $R_m$ by the time the counter reaches $n$. In this case we have correctly computed $p^{v+1}$ for all $v < u - 1$ and, so, for all requirements of priority less than $m$. We know by our assumptions that no extension will be found at stage $s$ for any requirement of higher priority than $m$. Thus, $p^{v+1}$ will be defined as $p^{v+1}$ for the largest $v < u - 1$ for which we have already defined it. At the substage of stage $s + 1$ at which we consider $R_m$, we will search over strings including $\sigma$ and will be in the same situation as we were in the previous case at stage $s$. We can, thus, calculate $C(\sigma)$ from $G_k$ using $p^{v+1}$ exactly as we did in the previous case using $p^\sigma$. $\square$

This concludes the proof that every finite lattice can be embedded in the $p$-$T$ degrees of the recursive sets. We now explain the modifications necessary to embed an arbitrary recursive lattice $\mathcal{L}$ in $R_{p-T}$. Our starting point is the recursive sequence of sub-usl's $\mathcal{L}_n$ with union $\mathcal{L}$ and of representations $\Theta_n$ for them as lattices given by Theorem 2.7. We fix some recursive procedure $T$ for calculating the sequences $\mathcal{L}_n, \Theta_n$ and $\Sigma_n$, the range of the elements of $\Theta_n$. All of the elements of these sets are now coded as numbers in binary notation. The function $g : \mathbb{N} \rightarrow \Theta$ that we are constructing conforms to the additional constraint for each $n$, $g(n) \in \Theta_m$, where $m$ is the largest $s$
such that we have completed the construction of $\mathcal{L}_s, \Theta_s$ and $\Sigma_s$ by step $n$ of the procedure $T$. Our notion of forcing $\mathcal{P}$ now consists of all finite initial segments $p$ of such functions. Our projection functions for each $j \in \mathcal{L}$ are defined with respect to the sequence of representations: $0^{m^1} \land \tau \in \mathcal{G}_j \iff j \in \mathcal{L}_m, g(n) \in \Theta_m$ and $(g(n))(i) = \tau$, where $m$ is as in the constraints on $g$ above. We define $P_j$ similarly for $p \in \mathcal{P}$. With these projections specified, the definition of an element $p$ of $\mathcal{P}$ forcing some basic sentence about $G_i, G_j$ and $G_k$ is as before. We should note that if we use these projections to define an embedding into $\mathcal{R}_{p-T}$ as for the finite case, zero, order and join are again automatically preserved. We simply repeat the proofs of Lemmas 2.3–2.5 with the understanding that we work inside the representation $\Theta_m$ determined by the $n$ of the string $\tau - 0^{m^1} \land \nu$ about which we are inquiring. The requirement that all of $\mathcal{L}_m, \Theta_m$ and $\Sigma_m$ have been computed in less than $n$ steps makes the entire calculation polynomial in $n$ and, so, in the length of $\tau$.

Our list of requirements is determined as before. For convenience we take care to spread them out (by padding with dummy requirements as necessary), so that $R_s$ deals with elements $i, j$ and, possibly, $k$ of $\mathcal{L}$ which are all in $\mathcal{L}_m$ as computed in at most $n$ steps of $T$. Formally, the construction is the same as in the finite case except for the appropriate notational changes. At stage $s$, we replace $\Theta$ by $\Theta_m$, where $m$ is determined by the length of the string we are considering adding to $p^s$ [this ranges from one more than the length of $p^s$ to the various values of $c(s, u)$]. The elements $\alpha, \beta$ of $\Theta$ are replaced by binary strings representing elements of $\Theta_m$. One should particularly note that when we specify a search over strings of length less than, say, $c(s, u)$, we mean over all appropriate strings with leading strings of zeros of length at most $c(s, u)$.

The proof that nonordering is preserved (Lemma 3.4) is the same as before. The infimum requirements also require little comment in addition to the obvious notational changes. The essential one is in the proof of Lemma 3.5: for each pair $p(v), s(v)$, we find interpolants $\gamma_1, \gamma_2$ and $\gamma_3$ in $\Theta_m$, where $m$ is determined by $v$ as usual.

We have, thus, completed the proof of our embedding theorem.

**Theorem 3.7.** Every recursive lattice can be embedded in the polynomial-time Turing degrees of the recursive sets.

For any set $D$, we let $\mathcal{R}^D_{p-T}$ be the $p-T$ degrees of sets recursive in $D$ and $p-T$ above $D$. We now apply our embedding result to get our promised nonhomogeneity corollary for these structures.

**Theorem 3.8.** If $\mathcal{R}^D_{p-T} \cong \mathcal{R}^E_{p-T}$, then $D \leq_{p-T} F^{(3)}$ and, of course, $E \leq_{p-T} D^{(3)}$ (Here $D^{(3)}$ is the triple Turing jump of $D$.) The same conclusion also holds if we instead assume that the $p-T$ degrees of all sets recursive in $D$ are isomorphic to those recursive in $E$.

**Proof.** First note that our embedding theorem relativizes. Every lattice $\mathcal{L}$ recursive in $D$ can be embedded in $\mathcal{R}^D_{p-T}$. For the required coding, note that in [32, pp. 256–258] it is shown that for any set $A$ there is a finitely generated lattice $\mathcal{L}_A$ recursive in $A$ such
that if $L_A$ can be embedded in an usl $L$, then $A$ is recursive in the jump of any presentation of $L$ even as a partial order. The final point is then that $R_{p,T}$ is presentable recursively in $D^{(2)}$. This is clear from the representation by natural numbers $e$ with the ordering given by $e \leq i \Rightarrow \varphi_i^e \leq_{p,T} \varphi_i^D \wedge D \exists j (\varphi_i^D \wedge D = \varphi_i^e)$. (If one is uncomfortable with multiple representatives for each element, one can simply take the least $e$ of each equivalence class. This remains recursive in $D^{(2)}$, although no longer in $\Sigma^0_2$.) This argument (with the “$\oplus D$” deleted) also establishes the same result under the second assumption of the theorem.  

The proof of this last theorem also shows that there are countable lattices which cannot be embedded in $R_{p,T}$ for $A$ not recursive in $\emptyset^{(3)}$. We can improve recursively presentable in our embedding result to presentable recursively in $\emptyset'$, but much more than that seems difficult. Indeed, it is not clear what sort of precise characterization could be given of those lattices which are embeddable in $R_{p,T}$.

We would also like to remark that the general format used for the construction here and in [29, 30] allows for more general types of actions by the requirements. In particular, it is not necessary that we either do nothing for each requirement or act so as to immediately satisfy it. A requirement $R_n$, for example, can try to guarantee some infinitary result such as arranging that all but finitely many strings of even length are in the set we are building or almost all are not in it depending on some recognizable fact occurring or not. In such a situation the requirement $R_n$ is said to impose an environment at each stage $s$ [all (or no) even length strings must go in] which must be respected by lower priority requirements when at the next stage they search for eligible extensions to satisfy their own needs. Such restrictions will be observed from some point on in the construction and will not impair our ability to recover the necessary information to verify the lemmas which prove that the other requirements are all satisfied as long as they themselves eventually want to impose some constant environment. Abstract definitions and examples of such constructions can be found in [29, 30].

We now finally state and prove our extension of embedding result. Remember that we wish to show that given any sentence of the form

$$\Gamma(x_1, \ldots, x_n, y_1, \ldots, y_m) = \forall x_1, \ldots, x_n \exists y_1, \ldots, y_m (\Theta(x_1, \ldots, x_n) \rightarrow \Psi(x_1, \ldots, x_n, y_1, \ldots, y_m))$$

with $\Theta$ a complete diagram (with 0) in the $x_i$ in which every pair of elements has a least upper bound (and, so, the partial ordering determined by $\Theta$ describes a lattice) and $\Psi$ a complete diagram (with 0) in the $x_i$ and $y_j$ such that $\Psi$ is consistent with the lattice structure determined by $\Theta$, is true in $R_{p,T}$.

**Theorem 3.9** (Extendibility). Consider any sentence $\Gamma$ as described above. Let $L$ with the operations $\lor$ and $\land$ be the lattice defined on the $x_i$ (and 0) by $\Theta$. We let $\leq$ be the
partial ordering on both the $x_i$ and the $y_k$ determined by $\Theta$ and $\Psi$. Suppose that $a_1, \ldots, a_n$ are any degrees in $\mathbb{R}_{p,T}$ satisfying $\Theta$. Suppose further that

1. if $y_k \leq x_i, x_j$, then $y_k \leq x_i \wedge x_j$, and
2. if $x_i, x_j < y_k$, then $x_i \vee x_j < y_k$.

We can then find degrees $b_1, \ldots, b_m$ in $\mathbb{R}_{p,T}$ such that $\Psi(a_1, \ldots, a_n, b_1, \ldots, b_m)$ is true in $\mathbb{R}_{p,T}$.

**Proof.** Let $A_1, \ldots, A_n$ be sets of degrees $a_1, \ldots, a_n$, respectively (necessarily $p$-T above $\emptyset$). For notational convenience we set $x_0 = 0$ and $A_0 = \emptyset$. We also introduce the symbol $x_{n+1}$, set it to be above all the $x_i$ and $y_j$ and choose a set $A_{n+1}$ of degree $a_{n+1}$ strictly above all the $a_i, i \leq n$. (All sets considered here will be subsets of $\{0,1\}^\ast$.) Conditions (1) and (2) now guarantee that there is, for each $y_k$, a greatest $x_i$ below it and a smallest $x_i$ above it. We call these elements $x_{g(k)}$ and $x_{s(k)}$, respectively.

Our plan is to construct recursive sets $C_k \leq_{p,T} A_{s(k)}$ and then to define $B_k = A_{g(k)} \oplus \{ C_j | y_j \leq y_k \}$. We will arrange our construction so that the $b_k = \deg(B_k)$ are the required degrees in $\mathbb{R}_{p,T}$. That any $b_k$ so defined satisfies the positive ordering facts of $\Psi$ is straightforward. (Note that $y_j \leq y_k$ implies that $A_{g(k)} \leq_{p,T} A_{g(j)}$ and $A_{s(j)} \leq_{p,T} A_{s(k)}$.) It, therefore, suffices to meet all requirements $R_i$ of the following forms:

$N_{e,j,k}:$ If $y_j \leq x_k$, then $\varphi^k_{e_0} \neq C_j$.

$P_{e,i,k}:$ If $x_i \leq y_k$, then $\varphi^k_{e_0} \neq A_i$.

$Q_{e,j,k}:$ If $y_j \leq y_k$, then $\varphi^k_{e_0} \neq C_j$.

(Note that the $N_{e,j,k}$ and $Q_{e,j,k}$ guarantee the desired inequalities as $C_j \leq_{p,T} B_j$ by definition.)

We meet these requirements via a typical delayed diagonalization construction. Our approximations (forcing conditions) are now $m$-tuples of finite sets of strings defining the sets $C_1, \ldots, C_m$ on all strings of length less than some $n$.

**Construction:** At each stage $s$ we have defined a function $c(s)$ and every $C_k$ on all strings of length less than $c(s)$. We also have already satisfied all requirements $R_i$ for $i < s$. Our action is now determined by the requirement $R_s$ that we wish to satisfy.

$R_s = P_{e,i,k}$: Note first that by definition $x_i \leq y_k$ implies $x_i \leq x_{g(k)}$. In this case we define all the $C_j$ to contain no strings of length $m$ for $c(s) < m \leq t$, where $t$ is the least number greater than $c(s)$ by which we can calculate (all that has gone before and) that there is a witness $\tau$ such that $\varphi^k_{e_0}(\tau) \neq A_i(\tau)$. (Note that all the $A_i$ are recursive and we assume some fixed computation procedure for them. This, combined with some standard search, determines the procedure by which we look for such a witness.) The point here is that $B_k$ consists of $A_{g(k)}$ plus some of the $C_j$. We have determined finitely much of the $C_j$ and now continue to keep them all empty until we hit strings whose length $t$ is large enough so that in $t$ steps we can compute all that we need to see the diagonalization witness. There must be such a witness since the procedure of adding no new elements to any $C_j$ threatens to make $B_k$ the same as $A_{g(k)}$ [except on strings of length less than
As \( x_i \preceq x(g(k)) \), \( A_i \not\preceq_{p-T} A(g(k)) \). We must, therefore, eventually find such a witness. We now let \( c(s+1) \) be this \( t \) and go on to the next stage.

\( R_s = Q_{e,j,k} \): Here we keep all strings of length \( >c(s) \) up to some \( t \) out of every \( C_i \) except for \( C_j \). For strings \( \sigma \) of length greater than \( c(s) \), \( \sigma \) is put into \( C_j \) if it is in \( A(s(j)) \). Again we keep this up until we have defined the \( C_i \) on strings of length \( t \) which is large enough so that in \( t \) steps we can compute (all that has gone before and) that there is a witness \( \tau \) such that \( \varphi_e^{B_k}(\tau) \neq C_j(\tau) \). We know that there will eventually be such a witness since we are threatening to make \( B_k \) the same degree as \( A(s(j)) \) while we are making \( C_j \) the same as \( A(s(j)) \). As \( y_j \preceq y_k \), \( A(s(j)) \not\preceq_{p-T} A(g(k)) \) and, so, we eventually get the desired witness. Again we set \( c(s+1) \) equal to the least such \( t \) and go on to the next stage.

\( R_s = N_{e,j,k} \): The procedure here is the same as for \( Q_{e,j,k} \) except that we substitute \( A_k \) for \( B_k \). The point here is that we eventually find a diagonalization witness since by our assumptions \( A(s(j)) \not\preceq_{p-T} A_k \).

**Verification:** The verifications are routine. We consider only the one of a type not appearing in the embedding argument.

**Lemma 3.10.** For each \( j < m \), \( C_j \preceq_{p-T} A(s(j)) \).

**Proof.** Consider any string \( \sigma \) of length \( n \). To calculate \( C_j(\sigma) \), we first find (by running our construction for \( n \) steps) the largest \( s \) such that \( c(s) < n \). We thus know \( R_s \), and that \( C_j(\sigma) \) is defined at stage \( s \). If \( R_s \) is not of the form \( N_{e,j,k} \) or \( Q_{e,j,k} \) for some \( e \) and \( k \), then we know that \( \sigma \) is not in \( C_j \). Otherwise, \( C_j(\sigma) = A(s(j)) \).

These constructions combined with the analysis of \( \forall \exists \) sentences prove our main theorem.

**Theorem 3.11.** The class of all sentences (in the language of partial orderings with least element 0) of the form \( \forall x_1 \ldots \forall x_n \exists y_1 \ldots \exists y_m \Theta(x_1, \ldots, x_n, y_1, \ldots, y_m) \), with \( \Theta \) quantifier-free, which are true in \( R_{p-T} \) is decidable.

## 4. Other reducibilities and further problems

One consequence of our results is the emphasis it places on the distinctions between \( p-T \) and \( p-m \) (polynomial-time many-one) reducibility introduced by Karp [1]. This difference was first established by Ambos-Spies: \( R_{p-m} \) is distributive but \( R_{p-T} \) is not. It is clear that our lattice representation coding scheme is a tt but not a many-one coding. Of course, this is necessary by the results of Ambos-Spies. Whether, or how, our general methods of analysis and construction might apply to \( R_{p-m} \) is, however,
unclear. As far as deciding the $\forall \exists$-theory of $R_{p,m}$, our general analysis still says that we need only consider sentences of the form

$$\Gamma(x_1, \ldots, x_n, y_1, \ldots, y_m) = \forall x_1, \ldots, x_n \exists y_1, \ldots, y_m (\Theta(x_1, \ldots, x_n) \rightarrow \bigvee_{i < h} \Psi_i(x_1, \ldots, x_n, y_1, \ldots, y_m)).$$

If the diagram $\Theta$ is one of a distributive lattice, then we are in the same situation as for $R_{p,T}$. Every finite distributive lattice can be embedded in $R_{p,m}$ [4] and, so, we need only check that our extension of embedding argument works for many-one degrees. The codings used there are all simple many-one reductions and, so, one only needs to check that the reduction given in Lemma 3.10 is a many-one procedure. This is clear since in a calculation polynomial in the length of the given $\sigma$ we see either that $\sigma \notin C_j$ or that $C_j(\sigma) = A_{d,j}(\sigma)$. The difficulty comes when $\Theta$ is a diagram for an usl that, considered as a lattice, is nondistributive as, for example, $M_1$ (Fig. 1). In this case no one lattice embedding in $R_{p,m}$ can handle all the disjuncts $\Psi_i$. We could have $\Psi_1$ asking for a $y$ above both $x_2$ and $x_4$ but not above $x_3$ (and, so, violating $x_2 \lor x_4 = x_1$), while $\Psi_2$ asks for ones below both $x_2$ and $x_4$ and below $x_3$ and $x_2$ (violating the implied infima). As both possibilities are realizable and no one realization of the $x$'s will rule out both, we cannot decide this sentence on the basis of lattice embedding and extension results alone. We can, however, solve the extension of embedding problem for $R_{p,m}$ based on these results. The analysis and its decisions are exactly as for the wtt-degrees of the r.e. sets as described in [8]. Deciding the extension of embedding problem in this case is not, however, sufficient to decide the $\forall \exists$-theory.

Another view of the nondistributivity question is suggested by the remark of Ambos-Spies [2] that the $p$.1-tt degrees of recursive sets are distributive, but for $n \geq 2$ the $p$.n-tt degrees are not. (These reducibilities correspond to limiting the number of queries in the polynomial tt-reduction to $n$ and were introduced by Ladner et al. [17].) Our coding scheme uses full tt-reductions when considered over all lattices, but for any particular lattice only a fixed number of questions depending on the size of the lattice representation used are needed. This raises the intriguing possibility that as $n$ increases, more and more lattices are embeddable in the $p$.n-tt degrees of the recursive sets.

We, as is common practice, have used the Turing degrees as our model for the $p$.T degrees sometimes looking at all of $\mathcal{D}_T$ and, at other times, at the substructures $\mathcal{D}_T(\leq 0')$ or $\mathcal{R}_T$. Turing reducibility is distinguished from stronger reducibilities by the fact that the questions to be asked of the oracle for a given input are determined during the computation rather than being recursively specified in advance as in all stronger reducibilities. By this standard, $p$.T reductions are really tt-reductions (albeit not polynomial-time ones). Our results and others suggest that the analogy with tt-degrees should be explored further. Another aspect of the time-bounded reducibilities that is crucial for these constructions is the existence of a recursive listing of the reducibilities (unlike the situation for Turing degrees). To put them on a more even
The $p$-T-degrees of the recursive sets

keel with tt-degrees, we should perhaps consider $\mathcal{P}_n (\geq 0')$ or $\mathcal{P}_n (\geq 0'')$, as we can then list the tt-reductions sufficiently effectively. Indeed, $\mathcal{P}_n (\geq 0')$ is dense [24]. This analogy should be investigated further.

The final question suggested to us by these results concerns definability in $\mathcal{D}_{p-T}$, the $p$-T degrees of all sets. Our extension of embedding constructions crucially depends on the fact that the embedding that we wish to extend is into the degrees of recursive sets. This suggests the possibility that the $p$-T degrees of the recursive sets might be definable as those below which one can always extend embeddings in this way. If so, it will be possible to define the property of being recursive on arbitrary sets solely in terms of relative computability via polynomial-time reductions. The related analysis for $\mathcal{D}_T$ has been used by Jockusch and Shore to show that the Turing degrees of the arithmetic sets are definable in $\mathcal{D}_T$. This result for $\mathcal{D}_T$ plays a key role in other definability results as well as in the results on automorphisms and homogeneity. Defining recursiveness in terms of p-time reducibility in addition to being an intrinsically interesting result might also lead to a deeper understanding of the overall structure of this reducibility.

References