Previous work has addressed the issues of idempotence and boundedness for various restricted classes of Horn-clause queries. In this paper, we consider queries consisting of a single clause containing a single predicate symbol. As such, these queries are a notational variant of the full, untagged tableau queries with recursive semantics. The study of the idempotence and boundedness for single-clause, single-predicate queries has previously been restricted to the typed case. We generalize these results to obtain syntactic, polynomial-time computable characterizations of idempotence for certain classes of untyped queries.

1. INTRODUCTION

For some time it has been recognized that so-called "relationally complete" [Codd72] query languages are inadequate for expressing a number of reasonable queries [AhU179]. This limitation has led to proposals for increasing the expressive power of relational query languages by allowing recursive queries to be formulated. As a result, there is currently widespread study on a number of issues concerning recursive queries, such as, their efficient evaluation (e.g. [BaRa86]) and the design of languages in which to express them (e.g. [CMW87]). In this paper we are interested in the more fundamental notion of recognizing which recursive expressions in fact denote non-recursive or first-order queries. This problem, known as bounded or data-independent recursion, has also been addressed in [Ioan85, CoKa86, Naug86a, NaSa87, Sagi88]. Apart from being of theoretical interest, this problem has practical significance for the optimization of recursive queries.

The bounded recursion problem has been studied for two classes of query languages, one based on linear recursive Horn clauses [Ioan85, Naug86a], and the other based on untagged recursive database tableaux [CoKa86, Sagi88], which are essentially single-predicate Horn-clauses. These two languages have incomparable expressive power. A linear recursive rule query allows many relations (predicates) to be referenced in the query, while a tableau query can refer to only a single relation. On the other hand, tableau queries permit any number of occurrences of the recursive predicate, thus admitting non-linear queries.
Most work on database tableaux has centred on the typed case, where no variable can appear in more than one column of a tableau. Effectively this means that one can consider all the attribute domains of the relation being queried to be disjoint. As a query language, however, typed tableaux are very limited. For example, a query as simple as computing the transitive closure of a relation requires an untyped tableau for its formulation.

A syntactic characterization of idempotent typed tableaux is given in [Sagi85]. The problem of k-boundedness, where a tableau need be composed with itself at most k times to compute all the answers to a given query on any relation, is studied in [CoKa86, Sagi88]. A necessary and sufficient condition for a typed tableau to be k-bounded is given in these papers. Studying the idempotence and boundedness problems for untyped tableaux seems considerably more difficult. For example, the general boundedness problem for untyped tableaux is still not known to be decidable [CoKa86].

In this paper, we make some progress towards finding a syntactic characterization for idempotent single-predicate Horn clauses or untyped tableaux. Specifically, we generalize typed tableaux to what we call semi-typed tableaux and study the idempotence problem for this class. The next section introduces the required background material and definitions. In Section 3, we define semi-typed tableaux and review some properties of typed tableaux that do not hold in general for semi-typed tableaux. We also present the syntactic characterization of idempotent typed tableaux. In Section 4, a number of necessary conditions as well as sufficient conditions for a semi-typed tableau to be idempotent are derived. In addition, we provide exact syntactic characterizations of idempotence for various subclasses of semi-typed tableaux. Some difficulties with generalizing these results to general untyped tableaux are outlined in Section 5. Section 6 contains suggestions for further research in the area.

2. BACKGROUND

We assume that the reader is familiar with the basic definitions of relational database theory as found, for example, in [Ullm83]. In what follows, we will restrict our attention to databases consisting of a single relation, and hence will use the terms relation and instance interchangeably.

Tableaux can be used to express queries over a given database scheme. Since our database scheme comprises only a single relation scheme, the tableaux we consider are those usually referred to as untagged tableaux. Let $R$ be a relation scheme defined over the attributes $A_1, \ldots, A_n$. A tableau $T$ is a table with columns $A_1, \ldots, A_n$, and rows that are tuples of variables (that is, we consider only full tableaux). Each row of $T$ must have variables in all of its columns, and no variable may repeat in a row. The rows of $T$ are partitioned into a special row $s$, called the summary of $T$, and rows $w_1, \ldots, w_m$ which comprise the body of $T$. The variables appearing in the summary of $T$, namely $s(A_1), \ldots, s(A_n)$, are called distinguished variables (dv's), while those that appear only in the body of $T$ are called nondistinguished variables (ndv's). In our examples, dv's will usually be denoted by $x, y, z, \ldots$, while ndv's will usually be denoted by subscripted b's. Unique ndv's will sometimes be represented by hyphens. An ndv which appears more than once in a tableau $T$ is called a shared ndv. Each dv must appear in some row in the body of $T$. If the dv $s(A)$ appears only in column $A$ and no ndv appears in more than one column of $T$, then $T$ is said to be typed. The class of all typed tableaux is denoted by $\tau$.

A tableau defines a mapping from instances to relations. Let $r$ be a relation over scheme $R$. A mapping $h$ is a valuation of $T$ into $r$ if $h$ maps every row of $T$ to a tuple of $r$. The tableau $T$ maps relation $r$ to the following relation, denoted $T(r)$,