97-QP

Query Processing in a Video Retrieval System*

K.L. Liu[1], P. Sistla[1], C. Yu[1], N. Rishe[2]
1: Department of EECS, University of Illinois at Chicago
Chicago, IL 60607
2: School of Computer Science, Florida International University,
Miami, FL 33199

Abstract

In an earlier paper, we designed a similarity based video retrieval system. Queries are specified in a language called Hierarchical Temporal Language (HTL). In this paper, we present several extensions of the HTL language. These extensions include queries that can have the negation operator and any other logical and temporal operators such as disjunction. Efficient algorithms for processing queries in the extended language are also presented.

1 Introduction

Multimedia information systems have received much interest in recent years. It is expected that a major part of image data such systems handle will be composed of video. The issue of how to retrieve the desired videos will be of practical importance. There has been some earlier work on modeling video data and specifying video queries [4, 5, 6, 9, 10, 19, 3, 12]. Most of them use exact matching for retrieval purposes. An important characteristic of our video retrieval system is that retrieval is similarity based, which is generally accepted as more appropriate for retrieval from multimedia databases than that based on exact matching. A major part of our system consists of methods for similarity based retrieval that are built on top of an existing picture retrieval system (described in [1, 2, 15, 16, 17]). In [18], we presented a hierarchical model for video databases, a Hierarchical Temporal Language (HTL) for specifying queries and efficient algorithms for various important subclasses of formulas of HTL. In this paper, we present a number of non-trivial improvements and extended the system so that it can process a wider class of HTL formulas.

The main contributions of this paper are as follows.

- Extensions to the Hierarchical Temporal Language.
- A more general efficient algorithm for processing HTL formulas involving any logical or temporal operator but not negation.
- An efficient algorithm for processing HTL formulas with negations.

*This research is supported by the following organizations: NSF grants under IRI-9309225 and IRI-9509253, NASA under NAGW-4080 and ARO under BMDO grant DAAH04-0024.

Our paper is organized as follows. Section 2 gives a brief review of the Hierarchical Temporal Language and the processing of HTL formulas. We show in section 3 how to modify our algorithm presented in [18] to become general enough to process all HTL formulas that have existential quantifiers but not negations. For formulas involving negations, their processing is discussed in section 4. Concluding remarks are given in section 5.

2 Review

The meta-data for the videos is represented using a hierarchical model. In this model, a single video is arranged into various levels. Each level consists of a temporally ordered sequence of video segments which is a decomposition of video segments at the next higher level. At the top level we simply have a single video segment representing the whole video. At the next level this video may be decomposed into a sequence of sub-plots, and each sub-plot may be further decomposed into a sequence of scenes at a lower level; at a still lower level, each scene may be decomposed into a sequence of shots. At the lowest level each shot is a sequence of frames. Meta-data is associated with each video segment at each level in the above hierarchy. The meta-data contains information about the objects in the video segments, their properties and the relationships among them.

The query language of our system is called Hierarchical Temporal Language (HTL). It is an extension of classical temporal logic of [11] and Future Temporal Logic [14]. This language uses the classical temporal operators to specify properties of video sequences (i.e. the temporal properties). In the following, we give some examples of operators in our HTL query language which we shall refer to in subsequent sections. Two of these examples, *FOLLOWED_BY* and *OR*, were not described in [18].

- $f = g$ *UNTIL* h : f is satisfied at video segment u if there is a video segment u' which is the same as u or which appears after u such that h is satisfied at u' and g is satisfied at all video segments between u and u' with a minimum threshold value.
- $f = g$ *FOLLOWED_BY* h : Formula f is satisfied at a segment u if g is satisfied at u

and h is satisfied at some segment which occurs after u.

- $f = g\ OR\ h$: f is satisfied at a video segment u, if either g is satisfied at u or h is satisfied at u.

Let $f = \exists X_1 X_2 \ldots X_n\ g(X_1, \ldots, X_n)$ be a formula with existential quantifiers. An *evaluation* ρ for f is a function that assigns values to the object variables $X_1, X_2, \ldots, X_n$; for example, $X_1 = 21, X_2 = 35, \ldots, X_n = 98$ is a possible evaluation, where X_i may denote the ID of some object. The similarity values of f at various video segments with respect to different evaluations are presented in a *similarity table*. This table is formed by inductively computing a similarity table for each subformula h of f. If subformula h has k object variables appearing free in it. The similarity table for h will have $k+1$ columns. The first k column names will be the names of the free variables appearing in h and the $(k+1)$st column will be a similarity list, which is a list of pairs giving the similarity values in different intervals of video segments. In each tuple, the values of the first k columns give an evaluation ρ for the formula, and the value of the last column is a similarity list denoting the similarity value of h at various video segments with respect to ρ. The following is an example of a similarity table T for the formula $\exists X_1 X_2\ h(X_1, X_2)$.

Table T

X_1	X_2	
1	3	(([2, 7], (8, 10)) ([18, 20], (2, 10)))
5	6	(([10, 15], (7, 10)))

Each entry in a similarity list in the last column of T is a pair $([u_1, u_2], (a, m))$, where the interval $[u_1, u_2]$ denotes a range of segments between u_1 and u_2 inclusive, a is the actual similarity and m is the maximum similarity.

The similarity table for a subformula h of f is inductively computed as follows. If h is non-temporal, then the table for h is computed using the approach given in [1]. When $h = h'\,op\,h''$, where op is a logical operator or a temporal operator, the similarity table T for h is computed in the following manner. Suppose the similarity tables for h' and h'' are T' and T'' respectively and the number of free variables in h', h'' and h is k', k'' and k respectively. The values in the first k columns of T are obtained by making a join of the first k' columns of T' with the first k'' columns of T'' where the join condition is the equality condition for the common column (i.e. variable) names. Let t' and t'' be tuples of T' and T'' respectively. Suppose the first k columns of tuple t of T are obtained by joining t' and t''. Then, the $(k+1)$st column value of t is obtained by combining the lists L' and L'' using the algorithm for the op operator, where L' and L'' are the lists in the last column of t' and t'' respectively.

Example 1 *Let* $f = \exists X_1 X_2 X_3 X_4\ \ g(X_1, X_2)$ *FOLLOWED_BY* $h(X_2, X_3, X_4)$. *Let* T_1 *and* T_2, *the similarity tables for subformulas* g *and* h *respectively, be given as follows.*

T_1

X_1	X_2	
11	*3*	(([2, 7], (8, 10))([18, 20], (2, 10)))
5	*6*	(([10, 15], (7, 10)))

T_2

X_2	X_3	X_4	
3	*9*	*23*	(([3, 8], (7, 15)))

Suppose T *is the similarity table for* f. *In column* X_2, *the value of the first tuple in* T_1 *is the same as the value of the tuple in* T_2. *These two tuples are joined to form a tuple* t *for* T. *The similarity list of* t, *formed by using the algorithm for the temporal operator FOLLOWED_BY, is* $(([2, 7], (7, 10)))$. *As the second tuple in* T_1 *and the tuple in* T_2 *have different values in column* X_2, *these two tuples are not joined. Hence, the similarity table* T *is*

Table T

X_1	X_2	X_3	X_4	
11	*3*	*9*	*23*	(([2, 7], (7, 10)))

■

3 Processing of HTL Formulas Without Negations

Suppose the operator in the formula of example 1 is not *FOLLOWED_BY*, but some other logical or temporal operator op. That is, $f = \exists X_1 X_2 X_3 X_4\ g(X_1, X_2)\ op\ h(X_2, X_3, X_4)$. Consider any video segment $u \in [10, 15]$. From table T_1 of example 1, with respect to the evaluation $X_1 = 5$ and $X_2 = 6$, the similarity of g at u is $(7, 10)$, greater than 0. As there is no tuple in T_2 which has value 6 in column X_2, the similarity of h at any segment, in particular segment u, with respect to any evaluation ρ in which $X_2 = 6$, must be 0. Note that for some operator op, such as the disjunction operator "OR", it is reasonable that the similarity of f at u is greater than 0 even if $h(X_2, X_3, X_4)$ is not satisfied at u. That is, with respect to any evaluation ρ in which $X_1 = 5$ and $X_2 = 6$, no matter what values X_3 and X_4 are assigned by ρ, the similarity of f at u is also greater than 0. Let N_{X_3} and N_{X_4} be the number of possible values of X_3 and X_4 respectively. Clearly, the number of different evaluations, in which $X_1 = 5$ and $X_2 = 6$ (or any other possible values), is the product $N_{X_3} \times N_{X_4}$; and the total number of different evaluations with respect to which formula f has non-zero similarity at some segment may be very large. A straightforward algorithm forms for the similarity table of f a tuple for each evaluation with respect to which f is satisfied at some segment. The number of possible evaluations such an algorithm need to consider in this case is $O(N_{X_1} N_{X_2} N_{X_3} N_{X_4})$. For a more complex HTL formula, the number of evaluations we need to consider could be extremely huge; and the execution of a straightforward algorithm may take an unreasonable length of time. For formulas that have negations, the problem is similar but only more severe. In this section, we show how this problem can be solved for HTL formulas without negations. For the case when negations are involved, the problem and its solution are addressed in the next section.

Note that with respect to an evaluation ρ, if a formula θ is not satisfied at all video segments (i.e. has similarity 0 at any segment), ρ is normally not represented by any tuple in the similarity table for θ. As we indicated above, for some binary operator op (e.g. the disjunction operator "OR"), with respect to an evaluation ρ, the similarity of $g\,op\,h$ may be non-zero at a video segment u even if one of its operands is not satisfied at any segment with respect to ρ. For such an operator, if the construction, described in the previous section or in [18], of a similarity table for $g\,op\,h$ is to work correctly, we must include in the similarity table for g (h) tuples to represent those evaluations with respect to each of which g (h) is not satisfied at all segments. However, as we showed in the previous paragraph, there may be a lot of such evaluations. If we include one tuple for each of these evaluations, the resulting table may be very large. For more complex (sub)formulas, their similarity tables may become unmanageable. To handle this situation, we introduce a notation $*$ to represent the set of **all possible values of an object variable**. For a collection of object variables X, Y, Z, for instance, the assignments $X = *$, $Y = 30$ and $Z = *$ denote the set of evaluations $S = \{(X, Y, Z) = (v_X, 30, v_Z) : v_X \in dom(X)$ and $v_Z \in dom(Z)\}$, where $dom(W)$ is the set of all possible values of variable W. To represent the set of all evaluations with respect to which h is not satisfied at any segment, we use a special tuple. This tuple has value $*$ for each object variable in h and an empty list in the column for similarity lists. We shall call this a *$*$-tuple*. In example 1, the $*$-tuple of T_2 is $(*, *, *, (\,))$. Note that as $*$ represents all possible values of an object variable, this $*$-tuple represents the set of all possible evaluations. However, we shall use it with the understanding that **it represents only those evaluations not represented by any other tuple** in the similarity table.

Throughout the following discussion, we refer to all formulas not involving any binary logical or temporal operators as *atomic* formulas and all other formulas as *compound* formulas. In the similarity tables for the atomic subformulas, we shall keep the *pointers* to the similarity lists rather than the actual similarity lists. We assume that distinct tuples of these similarity tables have distinct pointers; and we shall identify the tuples by their pointer values. In forming a similarity table T for a compound subformula h of a formula f, the similarity lists are not computed immediately. In each tuple of T, we keep the pointers to the similarity lists from which the similarity list of this tuple can be computed. As the similarity list of a $*$-tuple is empty, we shall indicate this by assigning the NULL value to its pointer column(s). Note also that in the following, **whenever the tuples of a similarity table are referred to, they do not include the $*$-tuple unless otherwise stated**; and the $*$-tuples will not be shown explicitly unless for clarity.

Let T_i be a similarity table for an atomic formula P_i and I_i the column in T_i for the pointers to similarity lists. Denote by $obj(P)$ the set of free object variables in formula P. Let h be a compound formula formed from n atomic formulas $P_i\,(i = 1, \ldots, n)$. The similarity table T for h has a column for each object variable in $\cup_{i=1}^{n} obj(P_i)$ and the columns $I_1, \ldots, I_n$. Each tuple t of T corresponds to a set of evaluations which are represented by the values in the columns corresponding to the object variables in $\cup_{i=1}^{n} obj(P_i)$. We denote this set of evaluations by S_t. $t.I_i$ is either from the set of values in column I_i of T_i or NULL. If $t.I_i$ equals $t_i.I_i$ for some tuple t_i in T_i, then $t.X$ equals $t_i.X$ for each $X \in obj(P_i)$; and at each video segment, the similarity of P_i with respect to any evaluation in S_t is the same and is given by the similarity list pointed to by $t.I_i$. If $t.I_i$ is NULL, the formation of tuple t involved the $*$-tuple of table T_i.

Definition 1 *Let t' and t'' be two tuples from similarity tables T' and T'' respectively. If for every common object variable X, $t'.X$ and $t''.X$ have the same value or one of them has the value $*$, the set of all possible values, we say that $S_{t'}$ and $S_{t''}$ are* **compatible** *or t' and t'' are compatible; we also say that the evaluations of $S_{t'}$ $(S_{t''})$ are compatible with t'' (t').* ∎

Suppose $h = h'\,op\,h''$, where op is a binary logical or temporal operator. Suppose further that h' is composed of the atomic subformulas $P_1, \ldots, P_w$; h'' composed of $P_{(w+1)}, \ldots, P_n$ and none of h' and h'' has negations. Let T' and T'' be the similarity tables for h' and h'' respectively. Tuples are joined based on the equality of the values of the common object variables. Evidently, only compatible tuples can be joined. Let $r' \in T'$ and $r'' \in T''$ be two compatible tuples and r a tuple resulting from joining r' and r''. For a common object variable X, if $r'.X = r''.X$, it is clear that $r.X = r'.X$ (or $r''.X$). Suppose one of $r'.X$ and $r''.X$, say $r'.X$, is $*$ which represents all possible values of X, and $r''.X = v$ $(\neq *)$. As only the value v among all the possible values of $r'.X$ equals $r''.X$, $r.X$ takes only the value v, i.e. $r.X = r''.X$. Thus, in the tuple r, the value for each $X \in obj(h') \cap obj(h'')$ is given by :

$$r.X = \begin{cases} r''.X & \text{if } r'.X = r''.X \text{ or } * \\ r'.X & \text{if } r''.X = * \end{cases}$$

Below, we present an algorithm, $NON\_NEG\_JOIN()$, for the construction of the similarity table T for $h = h'\,op\,h''$. The first two arguments of $NON\_NEG\_JOIN(T', T'', op)$ are the similarity tables for h' and h''. Step 1 of $NON\_NEG\_JOIN()$ is essentially the same as that described in [18]. Steps 2 and 3, if necessary, form those tuples representing evaluations with respect to which one of the operands of op is not satisfied at any video segment.

$NON\_NEG\_JOIN(T', T'', op)$

1. A table T is formed from tuples obtained by joining compatible tuples from T' and T''.

2. If a 0 similarity value of h'' at all segments may generate a non-zero similarity value for $h'\,op\,h''$

at some segments, then for each $t' \in T'$, t' is joined with the $*$-tuple of T'' and the resulting tuple is appended to T.

3. If a 0 similarity value of h' at all segments may generate a non-zero similarity value for $h'\,op\,h''$ at some segments, then for each $t'' \in T''$, t'' is joined with the $*$-tuple of T' and the resulting tuple is appended to T.

4. Join the $*$-tuples of T' and T'' to form the $*$-tuple of T.

5. Return T.

Let n' and n'' be the number of tuples in T' and T'' respectively. Let A be $obj(h') \cap obj(h'')$, the set of common object variables. The projection of T'' on A is $\prod_A T''$. Let W = set of distinct tuples in $\prod_A T''$. For a tuple $v \in W$, let m_v be the number of tuples in T' each having the same value as $v.X$ or the $*$ value in column X if $v.X \neq *$ for each $X \in A$; and n_v be the number of tuples in T'' each having the same value as $v.X$ in column X for each $X \in A$. Then, the number of tuples formed by joining the tuples in T' with those tuples in T'' is $\sum_{v \in W} m_v n_v$. Clearly, the total number of tuples formed in steps 2 and 3 is $O(n'+n'')$; and step 4 forms just the $*$-tuple. Hence, the size of the similarity table for $h'\,op\,h''$ is $\sum_{v\in W} m_v n_v + O(n'+n'')$.

Using hash join, step 1 of $NON\_NEG\_JOIN()$ requires time $O(\sum_{v\in W} m_v n_v)$. As the total running time of steps 2 and 3 is $O(n'+n'')$, the time complexity of $NON\_NEG\_JOIN()$ is $O(\sum_{v\in W} m_v n_v + n' + n'')$.

Example 2 *The similarity tables for the atomic subformulas $P_1(X_1, X_2)$, $P_2(X_2, X_3)$ and $P_3(X_3, X_4)$ are as follows. The values under the columns I_1, I_2 and I_3: 9, 10, 11, 21 and 15 are pointers to similarity lists.*

T_1

X_1	X_2	I_1
1	5	9
3	8	10

T_2

X_2	X_3	I_2
5	3	11
7	2	21

T_3

X_3	X_4	I_3
3	6	15

The similarity table T' for $P_1(X_1, X_2)\ UNTIL\ P_2(X_2, X_3)$ is

T'

X_1	X_2	X_3	I_1	I_2
1	5	3	9	11
$*$	5	3	NULL	11
$*$	7	2	NULL	21

From our definition of the $UNTIL$ operator (see section 2), it is clear that if the similarity of P_1 is zero at all segments (not satisfied at any segment), the similarity of $P_1\ UNTIL\ P_2$ may be non-zero at some segments (specifically, those segments at which P_2 is satisfied); if P_2 is not satisfied at all segments, $P_1\ UNTIL\ P_2$ cannot be satisfied at any segment. Hence, the tuples of T_2 are joined with the $*$-tuple of T_1 forming the second and third tuples of T' while the tuples of table T_1 are not joined with the $*$-tuple of T_2. The $*$-tuple of T' is ($*$, $*$, $*$, NULL, NULL).

The similarity table T for $(P_1(X_1, X_2)\ UNTIL\ P_2(X_2, X_3))\ OR\ P_3(X_3, X_4)$ is

X_1	X_2	X_3	X_4	I_1	I_2	I_3
1	5	3	6	9	11	15
$*$	5	3	6	NULL	11	15
1	5	3	$*$	9	11	NULL
$*$	5	3	$*$	NULL	11	NULL
$*$	7	2	$*$	NULL	21	NULL
$*$	$*$	3	6	NULL	NULL	15

Since the operator OR gives a zero similarity at a segment u only when both of its operands are not satisfied at u, the tuples of T' and the tuples of T_3 are joined with the $*$-tuple of T_3 and the $*$-tuple of T' respectively. ■

The $*$-tuple associated with a similarity table T is a special tuple. It has the value $*$ for each object variable and NULL values for the pointer columns. Although it represents the set of all possible evaluations, this $*$-tuple, as we mentioned above, is intended to represent only those evaluations not represented by any other tuple of T (because it may not be practical to enumerate all these evaluations).

Consider a tuple formed from a $*$-tuple of a table, say the second tuple $(*, 5, 3, 6, NULL, 11, 15)$, t_2, in T of example 2, which is formed from the $*$-tuple of T_1. The set of evaluations represented by t_2 is $\{(v_{X_1}, 5, 3, 6) : v_{X_1} \in dom(X_1)\}$, where $dom(X_1)$ is the set of all possible values of variable X_1. As is evident from its pointer columns, the other tuples from which t_2 is formed are the tuple $(5, 3, 11)$ of T_2 and $(3, 6, 15)$ of T_3, both of which are not $*$-tuples. From the I_2 and I_3 columns of T, it is clear that there is one other tuple, the first tuple $t_1 = (1, 5, 3, 6, 9, 11, 15)$ of T, which is also formed from the same tuples of T_2 and T_3; the tuple from T_1 to form t_1, however, is the tuple $(1, 5, 9)$, not the $*$-tuple. As an evaluation represented by $(1, 5, 9) \in T_1$ is not intended to be represented by the $*$-tuple of T_1, it follows that all the evaluations represented by t_1 are not intended to be represented by t_2. That is, the evaluation $(X_1, X_2, X_3, X_4) = (1, 5, 3, 6)$ is not intended to be represented by t_2. As there are no other tuples of T which are formed from the same two tuples of T_2 and T_3 from which t_2 is formed, the intended evaluations represented by t_2 is $\{(v_{X_1}, 5, 3, 6) : v_{X_1} \in dom(X_1)\} - \{(1, 5, 3, 6)\}$.

From the above discussion, we see that there are two sets of evaluations represented by each tuple t :

- A set of evaluations determined by the values in its variable columns. This set is denoted by $\mathbf{S_t}$. It may contain evaluations not intended to be represented by t. In the above example, $S_{t_2} = \{(v_{X_1}, 5, 3, 6) : v_{X_1} \in dom(X_1)\}$; it contains the evaluation $(1, 5, 3, 6)$ which is not intended to be represented by t_2.

- A set of intended evaluations represented by t, obtained by removing from S_t all evaluations not intended to be represented by t. We denote this set by $\mathbf{S_t^i}$. This is the set of evaluations actually represented by t. For the second tuple t_2 of T in example 2, $S_{t_2}^i = S_{t_2} - \{(1,5,3,6)\}$.

It should be clear that S_t^i is always a subset of S_t, and S_t and S_t^i may or may not be equal. Also, if a tuple t does not have any NULL pointer, it must be that $S_t^i = S_t$.

Once the similarity table $\mathcal{T}$ for a formula f has been formed, we then compute the similarity list for each tuple of $\mathcal{T}$. Let $\mathcal{P}_\mathbf{t}$ be the **set of non-NULL pointer values** of tuple t. Consider the first tuple t_1 of T in example 2. $\mathcal{P}_{t_1} = \{9,11,15\}$. The similarity list of t_1 will be computed from the three similarity lists pointed to by 9, 11 and 15 using the parse tree for the formula of example 2 and the algorithms in [18]. Note that the non-NULL pointer set of a tuple can be a subset of the non-NULL pointer set of another tuple. For instance, the non-NULL pointer set of the second tuple t_2 of T, $\mathcal{P}_{t_2} = \{11,15\}$ is a subset of $\mathcal{P}_{t_1}$. For these tuples, we have the the following proposition. (To save space, the proofs of all propositions are omitted.)

Proposition 1 *Let r' and r'' be two distinct tuples from the same similarity table. If $\mathcal{P}_{r'}$, the set of non-NULL pointer set of r', is a subset of $\mathcal{P}_{r''}$, then*

- *$S_{r''} \subset S_{r'}$; and*
- *all evaluations represented by r'' are not intended to be represented by r', i.e. $S_{r''} \cap S_{r'}^i$ is empty.*

In the above example, $\mathcal{P}_{t_2} \subset \mathcal{P}_{t_1}$ (t_1 and t_2 being the first two tuples of T of example 2). Evidently, $S_{t_1} = \{(1,5,3,6)\}$ is a subset of $S_{t_2} = \{(v_{X_1},5,3,6) : v_{X_1} \in dom(X_1)\}$; and we have shown earlier that all evaluations represented by t_1 are not intended to be represented by t_2.

Consider two distinct tuples r_1 and r_2 of a similarity table such that $\mathcal{P}_{r_1} \subset \mathcal{P}_{r_2}$. Since r_1 and r_2 are distinct, $\mathcal{P}_{r_1}$ must be a proper subset of $\mathcal{P}_{r_2}$. There must be a pointer column I, such that $r_1.I =$ NULL ($r_1.I$ points to an empty list) and $r_2.I \neq$ NULL. Let L_1 and L_2 be the similarity lists of r_1 and r_2 respectively. As the similarity functions for all the logical and temporal operators (except the negation operator, which we discuss in the next section) are monotonically increasing, the similarity value at any segment u given by L_1 cannot be greater than the similarity value at u according to L_2. As we want high similarities, the similarity list L_1 is not of much interest to us. Thus, for the similarity table $\mathcal{T}$ for f, we compute only the similarity lists of those tuples whose *non-NULL pointer sets are not proper subsets of any non-NULL pointer sets*; the similarity lists of the other tuples will not be computed unless specifically requested by the user. In table T of example 2, only the non-NULL pointer sets of the first and the fifth tuples are not the subsets of the non-NULL pointer sets of any other tuples. That is, only the similarity lists of these two tuples are needed to find those segments having highest similarity values. These tuples also have an interesting property which is stated in the following proposition.

Proposition 2 *Let t be a tuple in a similarity table T and $\mathcal{P}_t$ the set of non-NULL pointers of t. If for every $r \in T$, $r \neq t$, $\mathcal{P}_t$ is not a subset of $\mathcal{P}_r$, then all evaluations represented by t are intended to be represented by t, i.e. $S_t^i = S_t$.* ■

Consider the fifth tuple t_5 of T in example 2. $\mathcal{P}_{t_5}$ is not a subset of any other non-NULL pointer sets, and $t_5.I_1$ and $t_5.I_3$ are both NULL. It is readily seen that none of the evaluations in $S_{t_5} = \{(v_{X_1},7,2,v_{X_4}) : v_{X_1} \in dom(X_1)$ and $v_{X_4} \in dom(X_4)\}$ is compatible with any one of the tuples of T_1 and T_3. That is, $S_{t_5} = S_{t_5}^i$.

We have designed an algorithm to extract all tuples whose similarity lists contain high similarity values (i.e. those tuples whose non-NULL pointer sets are not proper subsets of any non-NULL pointer sets). Because of space limitation, the algorithm is not shown.

4 Processing of HTL Formulas Having Negations

The negation of a formula P is denoted by $\neg P$. If the similarity of P at segment u is (a,m), where a and m are the actual similarity and maximum similarity respectively, we compute the similarity of $\neg P$ at u as $(m-a,m)$. Suppose the similarity list for P is $L = (([1,5],(3,10)),([10,30],(8,10)))$. Note that any segment at which the similarity of P is 0 is not represented in L, and at such a segment the similarity of $\neg P$ is maximum according to our computation. Thus, if the set of all video segments is $[1,30]$, the similarity list L^c for $\neg P$ is $(([1,5],(7,10)),([6,9],(10,10)),([10,30],(2,10)))$.

Let h be an atomic formula consisting of object variables but not the negation operator. Let $\mathcal{T}_h$ be the similarity table for h, and I a column in $\mathcal{T}_h$ under which the pointers to similarity lists are stored. Consider the negation of h, $\neg h$. Let $\mathcal{T}_h^c$ be the similarity table for $\neg h$ and I^c the pointer column of $\mathcal{T}_h^c$. Corresponding to each tuple $t \in \mathcal{T}_h$, we form a tuple t^c for $\mathcal{T}_h^c$, which represents the same set of evaluations (i.e. $t.X = t^c.X$ for all $X \in obj(h) = obj(\neg h)$). The similarity list of t^c is computed from the similarity list of t in the same manner as we computed L^c in the previous paragraph. However, for efficiency, we do not perform this computation until the list is needed. We simply set the value of $t^c.I^c$ to point to the *same* similarity list of t. That is, $t^c.I^c = t.I$. Hence, t and t^c are equal. In other words, $\mathcal{T}_h^c$ and $\mathcal{T}_h$ have the same tuples. We simply set $\mathcal{T}_h^c$ to point to table $\mathcal{T}_h$ without forming $\mathcal{T}_h^c$ explicitly. This takes constant time. Let S_* be the set of all those evaluations not represented by any tuples of $\mathcal{T}_h$. In the previous section, we use a special $*$-tuple to represent the evaluations of S_*. The similarity list of the $*$-tuple is empty, indicating that h is not satisfied at any segment with respect to each evaluation in S_*. Hence, with respect to each $\rho \in S_*$, the similarity list of $\neg h$ has *maximum similarity for each video segment*; we represent such a similarity list by a special symbol U and call it a *U-list*. Normally, if with respect to an evaluation ρ, the similarity of a formula f is not 0 at some segment, a tuple will be formed to represent ρ. However, there are too many evaluations in S_*. To represent each of them by a tuple in a table is impractical. Furthermore, with respect to each of these evaluations, the similarity at each segment is the same. Hence, we choose to represent them using a single tuple. Since it represents the same set of evaluations as the $*$-tuple of $\mathcal{T}_h$, the variable columns of this tuple are assigned the same values as those in the corresponding columns of the $*$-tuple of $\mathcal{T}_h$, i.e. the $*$ values; its pointer column is assigned the value U. As this tuple serves a similar purpose as the $*$-tuple of $\mathcal{T}_h$, namely to *represent all evaluations not represented by any tuple of* $\mathcal{T}_h^c$, we call this the **$*$-tuple of** $\mathcal{T}_h^c$.

Consider the compound formula $h = h_1\ op\ \neg h_2$, where h_1 and h_2 are atomic formulas, both containing object variables. Let T_1, T_2^c and T be the similarity tables for $h_1, \neg h_2$ and h respectively; I_1 and I_2^c the pointer columns of T_1 and T_2^c respectively; and t_{*_1} and $t_{*_2}^c$ the $*$-tuples of T_1 and T_2^c respectively. The $*$-tuple of T representing all evaluations not represented by any tuple of T is formed by joining t_{*_1} and $t_{*_2}^c$. Note that $t_{*_2}^c.I_2^c$ equals U, i.e. the similarity list of $t_{*_2}^c$ is an U-list that has maximum similarity value for every video segment, and the similarity list of t_{*_1} is a $NULL$-list (i.e. empty). Depending on the binary operator op, the resulting similarity list of the $*$-tuple of T may be a $NULL$-list, an U-list or neither. If the similarity list is *not* a $NULL$-list or an U-list, we shall call it a *M-list*. To indicate what kind of similarity list the $*$-tuple has, we associate with each similarity table T a field *Flag* and refer to it as $T.Flag$. Let L_* be the similarity list of the $*$-tuple of T. The possible values of $T.Flag$ are

- N - indicating that L_* is the $NULL$-list (an empty list);
- U - indicating that L_* is an U-list that has maximum similarity value for every video segment;
- M - indicating that L_* is a M-list (i.e. not a $NULL$-list or an U-list).

Let L and L^c be two similarity lists for formulas P and $\neg P$ respectively. Clearly, if L is a $NULL$-list, then L^c is an U-list; if L is an U-list, then L^c is a $NULL$-list; and if L is a M-list, then L^c is also a M-list. Let T and T^c be two similarity tables for formulas h and $\neg h$ respectively. Thus, the relation between the values of $T.Flag$ and $T^c.Flag$ is given by the following table:

Table 1

$T.Flag$	$T^c.Flag$
N	U
U	N
M	M

As before, the similarity lists of a similarity table for a compound formula are not computed immediately. Thus, if h is a compound formula, the similarity table for $\neg h$ can be obtained from T by simply changing the *Flag* field according to the above table.

Let $h = h'\ op\ h''$, and T' and T'' the similarity tables for h' and h'' respectively. When one of the two operands of op, say h', consists of the negation operator, the $*$-tuple of T' may have an U-list, M-list or NULL-list. If the similarity list of the $*$-tuple is not a NULL similarity list, the $*$-tuple of T' needs to join with every tuple of T'' in the construction of the similarity table for h. Thus, we need to modify the algorithm $NON\_NEG\_JOIN()$, given in the previous section, for the construction of the similarity table for a compound formula. The modified algorithm, $GEN\_JOIN()$, is presented below. Steps 2 and 3 of $NON\_NEG\_JOIN()$ are changed so that they check first whether the similarity lists of the $*$-tuples of T' and T'' are NULL lists or not by examining the *Flag* field. We also include a step that determines the *Flag* field of the table for h (which takes constant time). Clearly, $GEN\_JOIN()$ has the same time complexity as $NON\_NEG\_JOIN()$.

```
GEN_JOIN($T'$, $T''$, $op$)

1. A table $T$ is formed from tuples obtained by joining tuples of $T'$ with compatible tuples of $T''$.
2. If • $T''.Flag \neq N$, or
      • $T''.Flag = N$ and a 0 similarity value of $h''$ at all segments may generate a non-zero similarity value for $h'\ op\ h''$ at some segments,
   then for each $t' \in T'$, $t'$ is joined with the $*$-tuple of $T''$ and the resulting tuple is appended to $T$.
3. If • $T'.Flag \neq N$, or
      • $T'.Flag = N$ and a 0 similarity value of $h'$ at all segments may generate a non-zero similarity value for $h'\ op\ h''$ at some segments,
   then for each $t'' \in T''$, $t''$ is joined with the $*$-tuple of $T'$ and the resulting tuple is appended to $T$.
4. Form the $*$-tuple of $T$ by joining the $*$-tuples of $T'$ and $T''$. Check the operator $op$ to determine whether the similarity list of the $*$-tuple is a NULL-list, $U$-list or $M$-list, and set $T.Flag$ to an appropriate value.
5. Return $T$.
```

Example 3 *Let* $h(X_1, X_2, X_3, X_4, X_5)$ *be the compound formula* $\exists X_1 X_2 X_3 X_4 X_5\ (\neg P_1(X_1, X_2)\ OR\ P_3(X_2, X_3))\ UNTIL(\neg P_2(X_3, X_4)\ FOLLOWED\_BY\ P_4(X_4, X_5))$. T_1^c, T_3, T_2^c *and* T_4, *the similarity tables for* $\neg P_1, P_3, \neg P_2$ *and* P_4 *respectively, are given as follows:*

Table T_1^c

X_1	X_2	I_1^c
9	10	1

$T_1^c.Flag = U;\ \ *\text{-tuple} = (*, *, U)$

Table T_3

X_2	X_3	I_3
15	6	2

$T_3.Flag = N$; $*$-*tuple* $= (*, *, NULL)$

Table T_2^c

X_3	X_4	I_1^c
6	12	3

$T_2^c.Flag = U$; $*$-*tuple* $= (*, *, U)$

Table T_4

X_4	X_5	I_4
12	8	4
2	3	5

$T_4.Flag = N$; $*$-*tuple* $= (*, *, NULL)$

The similarity table T' *for* $\neg P_1\ OR\ P_3$ *is*

Table T'

X_1	X_2	X_3	I_1^c	I_3
9	10	$*$	1	NULL
$*$	15	6	U	2

$T'.Flag = M$; $*$-*tuple* $= (*, *, *, U, NULL)$

$T_1^c.Flag = U$ means that with respect to any evaluation represented by the $*$-tuple of T_1^c, the similarity of $\neg P_1$ at any segment is the maximum similarity. As $T_3.Flag = N$, the similarity list of the $*$-tuple of T_3 is empty. According to our similarity function for the disjunction operator OR if at a segment u, the similarity of h_1 is maximum and the similarity of h_2 is 0, the similarity of $h_1 OR h_2$ at u is neither 0 nor the maximum similarity value. Thus, the similarity list of the $*$-tuple of T', which is neither a NULL-list nor an U-list, is a M-list. Hence, $T'.Flag = M$.

The similarity table T'' of $\neg P_2\ FOLLOWED\_BY\ P_4$ is

Table T''

X_3	X_4	X_5	I_2^c	I_4
6	12	8	3	4
$*$	12	8	U	4
$*$	2	3	U	5

$T''.Flag = N$; $*$-*tuple* $= (*, *, *, U, NULL)$.

If the second operand h_2 of $h_1\ FOLLOWED\_BY\ h_2$ is not satisfied at any segment, $h_1\ FOLLOWED\_BY\ h_2$ cannot be satisfied at any segment. Hence, the $*$-tuple of T_4, which has an empty similarity list, is not joined with the tuples of T_2^c; and the similarity list of the $*$-tuple of T'', formed by joining the $*$-tuples of T_2^c and T_4, must also be empty; that is, $T''.Flag$ is also N.

The similarity table T for $h = (\neg P_1\ OR\ P_3)\ UNTIL(\neg P_2\ FOLLOWED\_BY\ P_4)$ is

Table T

X_1	X_2	X_3	X_4	X_5	I_1^c	I_3	I_2^c	I_4
9	10	6	12	8	1	NULL	3	4
9	10	$*$	12	8	1	NULL	U	4
9	10	$*$	2	3	1	NULL	U	5
$*$	15	6	12	8	U	2	3	4
$*$	15	6	12	8	U	2	U	4
$*$	15	6	2	3	U	2	U	5
$*$	$*$	6	12	8	U	NULL	3	4
$*$	$*$	$*$	12	8	U	NULL	U	4
$*$	$*$	$*$	2	3	U	NULL	U	5

$T.Flag = N$;
$*$-*tuple* $= (*, *, *, *, *, U, NULL, U, NULL)$

Recall that if the similarity of h_2 is not satisfied at any segment, $h_1\ UNTIL\ h_2$ cannot be satisfied at any segment. As $T''.Flag = N$ (i.e. the similarity list of the $*$-tuple of T'' is empty), the value of $T.Flag$ must also be N. ■

For tuples of a similarity table for a formula with negations, we have two propositions which are almost identical to propositions 1 and 2 except that the "non-NULL pointers" in these propositions should be changed to "non-NULL and non-U pointers". We state only the one that corresponds to proposition 1 below.

Proposition 3 *Let r' and r'' be two distinct tuples from the same similarity table. Let $\mathcal{P}_t$ be the set of non-NULL and non-U pointer set of tuple t. If $\mathcal{P}_{r'}$ is a subset of $\mathcal{P}_{r''}$, then*

- *$S_{r''} \subset S_{r'}$; and*
- *all evaluations represented by r'' are not intended to be represented by r', i.e. $S_{r''} \cap S^i_{r'}$ is empty.*

Consider the similarity table T^c for the negation of an atomic formula. The similarity list of the $*$-tuple of T^c has maximum similarity values for all video segments, which is indicated by having the value U in the pointer column. Thus, a tuple having more U pointer values is likely to have higher similarity values in its similarity list. However, this also means that more evaluations are not intended to be represented by that tuple and it is even possible that there is no intended evaluation represented by that tuple. Consider the fifth tuple $t_5 \doteq (*, 15, 6, 12, 8, U, 2, U, 4)$ of T in example 3. No tuple in the same table has more U values than t_5 does. The set of non-NULL and non-U pointers of t_5, $\mathcal{P}_{t_5} = \{2, 4\}$, which is a subset of $\mathcal{P}_{t_4} = \{2, 3, 4\}$, the set of non-NULL and non-U pointers of t_4, the fourth tuple of T. Hence, by proposition 3, S_{t_4} is a subset of S_{t_5}, the set of evaluations represented by t_5. By the same proposition, no evaluation represented by t_4 is intended to be represented by t_5, i.e. $S^i_{t_5}$ can only contain evaluations from $S_{t_5} - S_{t_4}$. As t_4 and t_5 also have identical values in the variable columns, both t_4 and t_5 represents the same set of evaluations (i.e. $S_{t_5} = S_{t_4}$). Hence, $S_{t_5} - S_{t_4}$ is empty, which means that $S^i_{t_5}$ is also empty. Thus, there is no intended evaluation represented by t_5 even though its similarity list potentially has high similarity values.

From the above discussion, before we generate the similarity lists for the tuples of a similarity table T, we must first search for all tuples for which there are no intended evaluations represented by them and delete them from the table. We have designed an algorithm that identifies all such tuples by computing for each tuple t, S^i_t, the number of intended evaluations represented by t. We have also developed an algorithm to remove any tuple t in T such that at any video segment u, the value in the similarity list of

t is no greater than that in the similarity list of some other tuple in T. Because of space limitation, both algorithms are not shown.

5 Conclusion

Although the algorithm in our earlier work can process efficiently several important subclasses of HTL queries involving existential quantifiers [18], there are certain common queries that it is not well-suited to handle. Extensions, involving negation and disjunction, have been made to our HTL query language. We developed efficient algorithms for processing video queries in the extended HTL language; these algorithms are modified from our earlier algorithm. For a formula f that involves existential quantifiers and some common operators, such as disjunction "OR" and negation "$\neg$", the number of tuples that need to be formed for the similarity table for f by a straightforward algorithm may be prohibitively large. In comparison, the number of tuples formed for f by the modified algorithms is modest and does not depend on the number of possible values the variables in f can have.

Let $h = h'\ op\ h''$ be a subformula of f which involves existential quantifiers. Let T' and T'' be the similarity tables of h' and h'' respectively. Let n' and n'' be the number of tuples in T' and T'' respectively. Let A be $obj(h') \cap obj(h'')$, the set of common object variables. The projection of T'' on A is $\prod_A T''$. Let W = set of distinct tuples in $\prod_A T''$. For a tuple $v \in W$, let m_v be the number of tuples in T' each having the same value as $v.X$ or $*$ in column X if $v.X \neq *$ for each $X \in A$ ($*$ in column X represents all possible values of X.); and n_v be the number of tuples t in T'' such that $t.X = v.X$ for each $X \in A$. The time of our algorithm to form the similarity table of h is $O\Big(\sum_{v \in W} m_v n_v + n' + n''\Big)$. The number of tuples formed for h is $\sum_{v \in W} m_v n_v + {}$ $+ O(n' + n'')$.

References

[1] A. Aslandogan, C. Thier, C. Yu, et al, *Implementation and Evaluation of SCORE (A System for COntent based REtrieval of Pictures)*. IEEE Data Engineering Conference, March 1995.

[2] A. Aslandogan, C. Thier, C. Yu, et al, *Using Semantic Contents and WordNet^{TM} in Image Retrieval*, ACM SIGIR Conference, to appear in 1997.

[3] C. Breiteneder, S. Gibbs and D. Tsichritzis, *Modeling of audio/video data*, Proc. ER conference, 1992.

[4] T. Chua and L. Ruan, *A video retrieval and sequencing system.* ACM Transactions on Information Systems, October 1995.

[5] Y. Day, S. Dagtas, M. Iino, A. Khokhar and A. Ghafoor, *Object-oriented conceptual modeling of video data*, IEEE Data Engineering, 1995.

[6] N. Dimitrova and F. Golshani, *RX for semantic video database retrieval*, ACM Multimedia conference, 1994.

[7] A. Gupta, *Visual Information Retrieval Technologies, A Virage perspective*. White paper, Virage Incorporated, 1995.

[8] A. Hampapur, R. Jain and T. Weymouth., *Production Model based Digital Video Segmentation*. Multimedia Tools and Applications, Vol. 1, pp. 1-38, 1995.

[9] R. Hjelsvold and R. Midtstraum, *Modeling and querying video data*, VLDB 94.

[10] E. Hwang, V.S. Subrahmanian, *Querying Video Libraries*, Technical Report, Department of Computer Science, University of Maryland 1995.

[11] Z. Manna, A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems* Specification, Springer-Verlag 1992.

[12] E. Oomoto and K. Tanaka, *OVID: Design and implementation of a video-object database system*, IEEE TKDE 1993.

[13] I.K. Sethi and R. Jain, *Finding Trajectories of feature points in a monocular image sequence*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol 9, pp. 56-73, 1987.

[14] A.P. Sistla, O. Wolfson, *Temporal Triggers in Active Database Systems*, IEEE TKDE, July 1995.

[15] A.P. Sistla, C. Yu and R. Hadda, *Reasoning About Spatial Relationships in Picture Retrieval Systems*. VLDB, 1994.

[16] A.P. Sistla, C. Yu, C. Liu and K.L. Liu, *Similarity based retrieval of pictures using indices on spatial relationships*. VLDB, 1995.

[17] A.P. Sistla and C. Yu, *Retrieval of Pictures based on Approximate Matching*. Chapter in the book "Multimedia Database Systems", edited by V.S. Subrahmanian and S. Jajodia, Springer 1996.

[18] A.P. Sistla, C. Yu and R. Venkatasubrahmanian, *Similarity Based Retrieval of Videos*. International Conference on Data Engineering, Birmingham, U.K., 1997.

[19] R. Weiss, A. Duda, D.K. Gifford, *Composition and Search with Video Algebra*, IEEE Multimedia, pp 12-25, 1995.