Mexitl: Multimedia in Executable Interval Temporal Logic[⋆]

Howard Bowman[1], Helen Cameron[2], Peter King[2] and Simon Thompson[1]

[1] Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent, CT2 7NF, United Kingdom

[2] Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, R3T 2N2, Canada

Email Contacts: {H.Bowman,S.J.Thompson}@ukc.ac.uk and {prking,hacamero}@cs.umanitoba.ca

Abstract. This paper explores a formalism for describing a wide class of multimedia document constraints, based on an interval temporal logic. We describe the requirements on temporal logic specification that arise from the multimedia documents application area. In particular, we highlight a canonical specification example. Then we present the temporal logic formalism that we use. This extends existing interval temporal logic with a number of new features: actions, framing of actions, past operators, a projection-like operator called filter and a new handling of interval length. A model theory, logic and satisfaction relation are defined for the notation, a specification of the canonical example is presented, and a proof system for the logic is introduced.

1 Introduction

This paper explores a formalism for describing a wide class of multimedia document constraints. The formalism is based on an interval temporal logic. The term *multimedia* indicates that a document may contain continuous or time-dependent entities [10] known as *media items* [6], in addition to the more traditional spatial items in paper documents. Part of the task facing the author of such a document, therefore, is to describe the dynamic temporal relationships that are to hold between media items. We are interested in documents with rich sets of such relationships, and as a consequence are keenly interested in issues of consistency verification, modelling, proto-typing, and specification refinement. While a number of authoring systems for multimedia documents are extant [3, 9], little work has been done on the question of suitable formalisms for the specification and manipulation of such temporal constraints in the context of multimedia.

This paper introduces an interval temporal logic specification notation called *Mexitl* which stands for *Multimedia in Executable Interval Temporal Logic*. While

[⋆] Travel grants to support the research presented here have been provided by the British Council. The second and third authors are supported by individual research grants from the Natural Sciences and Engineering Research Council of Canada.

the anticipated application area for this notation is multimedia, it is also relevant to other areas of real-time specification. King [11, 12] proposed the use of Interval Temporal Logic (ITL) to specify multimedia documents and the theory presented here stems from that earlier work.

A major difference between ITL and standard linear time temporal logic [15], is that ITL is interpreted over finite state sequences, called intervals, rather than over infinite models. A number of authors have investigated ITLs, including [4, 8, 13, 16]. The restriction to finite states prompts consideration of a number of temporal operators not typically found in non-interval temporal logics, such as *chop* and *projection*, which will be discussed in Section 3.1.

We anticipate that complete specifications of multimedia documents will have a number of elements. Firstly, an *abstract data typing* notation will be used in order to describe the primitive *operations/actions* (we use these terms interchangeably in the sequel) to be used in specifying the document. Actions such as *displayCaption* or *playVideo* are typical examples. A suitable abstract data typing notation is suggested in [11]. We will not consider this notation here, rather the specification language that we present takes the primitive actions as given. In addition, mechanisms to define composite actions out of primitive actions can be added.

We introduce a methodology for developing multimedia artefacts using *Mexitl*. Specifications are written in the logic, and refined according to the rules of the language. Implementations can be developed in various ways, including either by means of deterministic refinements or as proofs of *Mexitl* formulas interpreted in a constructive logic.

Structure of the Paper. The paper is structured as follows. Section 2 reviews the requirements associated with multimedia documents. In particular, an example of a typical presentation from the field is highlighted. Section 3 presents the specification notation that we are advocating. The operators of the core language are presented, the model theory is highlighted and the satisfaction relation is defined. Section 4 contains an account of how the operators used in the paper are derived from the core operators of the language. Then Section 5 applies the defined notation to the requirements highlighted in Section 2. Section 6 gives an initial account of the logic of the *Mexitl* operators. Related work is discussed in Section 7 and concluding remarks are presented in Section 8.

Note that a shorter version of this paper has been published as [2].

2 Multimedia Documents

2.1 Multimedia Document Requirements

There is no extant formal or standard taxonomy of functional requirements for temporal constraints among media items in multimedia documents. However, the multimedia literature displays a good deal of commonality and agreement in this regard. Indeed, in [6], Erfle presents a set of eighteen issues, or functional requirements, which he regards as being sufficient to describe multimedia documents. This set was obtained by a study of what is provided in a number of existing authoring systems and standards. In [11], King presents an alternative set of eight requirements, which encompasses Erfle's set. We will not repeat this work in any detail here, but for the sake of completeness we will provide a summary of the requirements. We will divide our summary into two sets of requirements, a set of three general requirements, which are dictated mainly by the authoring aspect of this area of application, followed by eight individual functional requirements.

This summary of requirements will be illustrated by a fairly complex example. In Section 5.2, we will show how each component of this example is expressible in the formalism to be described in Sections 3 and 4.

General Requirements The most basic requirement is to represent the *display* of a media item. Indeed, we require various forms of display. We first require what we might term *standard* display, where a media item is simply presented in its normal fashion at its normal rate. In addition, to the extent that the physical properties of the medium in question permit them, we must allow in our formalism for variations of this standard display such as presenting at half speed, rewind, fast-forward, and so forth.

Secondly, we need to represent the notion of *reader intervention* during a multimedia display. This intervention will frequently require *early termination* of a media item.

Thirdly, we require facilities for *composing* sets of constraints. Documents, just like programs, are rarely static or fixed, and new documents are often created by adding to existing ones. An author will frequently use a section of an existing document as part of a new, larger document. We thus require the ability to express both serial and parallel composition of sets of media item constraints. Parallel composition is also required to permit independent development of what are termed *channels* in the multimedia literature [3, 9], which may then be combined so that they occur in the same time interval – that is, in the same multimedia presentation.

In this work we have chosen to generalise somewhat the notion of *channel* as it is used in the multimedia literature. See [3, 9], for example, where the term channel refers to a single type of medium, and one may refer, for example, to the video channel or the audio channel. By way of contrast, our notion of channel is a higher-level one, corresponding more closely to the notion of an independent author. In our work, therefore, a channel will usually contain items of differing physical media types, since a single author may well wish to make use of such a variety of media types in creating part of a complete multimedia document. Our use of the term channel is akin to the use of such terms as "thread" in other areas of computer science.

Functional Requirements The following eight individual items are required, although arbitrary combinations of these items are also to be permitted:

1. Temporal placement of a media item at an absolute (time) point;
2. Specification of the duration of a media item;
3. Determination of the start and finish points of a media item;
4. Relative placement of two or more media items;
5. Repetitive display of a media item;
6. Conditional display of a media item;
7. Scripting, that is, using events or conditions occurring in one media item to control the display of a second; and
8. Exception handling, that is controlling error situations which may occur during the display of a multimedia document.

2.2 Example

We will now illustrate the requirements list given in the previous section by introducing an informal specification of a fairly elaborate example of a multimedia document. We will annotate the various parts of this example with references to the corresponding items in the list given above. This entire example will be expressed in the ITL formalism in Section 5 below. We refer to this example as the *Beethoven Problem*. The Beethoven Problem requires the development of a multimedia document consisting of an audio of Beethoven's Fifth Symphony, Opus 67 in C minor, together with various other media items which are designed to illustrate the music as it is played. King [12] presents an earlier version of a portion of this example.

Beethoven's Fifth Symphony comprises four movements.

1. Play the four movements of the symphony in sequence with a gap of 20 seconds between each movement.
   This simple example uses sequential composition and duration specification; it involves Requirements 2 and 4. Note that King [11] shows in detail how n-ary temporal relations can be represented in (a subset of) the formalism described herein; we will not repeat the details of this. Strictly speaking, this example also involves Requirement 1, since we are assuming that the symphony is to begin at time zero. Note also that in what follows we will not be very concerned about temporal units or the granularity of real time. We may, for example, simply assume that one clock tick equals one second, or any other convenient unit.
2. Before the first movement begins, play an audio which announces the name of the symphony, the composer, and the orchestra. Two seconds after this audio starts, display a video still of Beethoven. Stop the video still display as the first movement starts. After the last movement, wait 3 seconds and display for 30 seconds a video of Ludwig van Beethoven and then, after a further 5 seconds, display information about how to order this presentation.
   The various parts of this specification may be composed in a variety of ways. For example, each might occupy a separate channel and all channels then be composed in parallel. Alternatively, they may be incorporated by

serial composition into one of the existing channels. The choice to be made is largely a matter of taste. As individual items, they involve serial composition together with Requirements 2 and 4.

3. At the start of each movement display an appropriate title for 5 seconds. Each title is a video still. Repeat the 5 second display of each title every 3 minutes during the corresponding movement.

   We may regard the display of the four video stills as comprising a second channel, which is to be composed in parallel with the first: the four movements. We have an instance of Requirement 5, a repetitive display. The actual display of the video stills themselves requires duration specification as in Requirement 2.

4. The audio introduced in Example 2 is actually in three parts, corresponding to the name of the symphony, the composer, and the orchestra. During this audio display,

   either display a rolling text item containing the same information (assume this item has the same length as the audio)

   or display, in sequence, three video stills containing the same information for the appropriate time lengths.

   This example is a further instance of parallel composition, but also introduces a conditional specification (Requirement 6), and also involves some simple sequential composition.

5. The twenty second gap between the second and third movements should be replaced by a video/audio display describing the third movement. If the video/audio display takes longer than twenty seconds, truncate the display to twenty seconds.

   This example illustrates the need for an exception handling facility; we must allow for truncation of the display in question if it does in fact require longer that 20 seconds, and decide what happens if it requires less that 20 seconds. Note, therefore, that we require the start and finish points of the item, an illustration of Requirement 3.

6. At the first sound of a viola, display a purple band for 3 seconds.
7. During the first crescendo passage, display a mammoth.
8. During each crescendo passage of the first movement, display a looped video tape of a bug climbing an inclined plane. For each crescendo passage after the first one, continue playing the video from the point at which it was previously stopped.
9. Count the number of C minor chords in the symphony.
10. During each staccato note of the first movement, flash the screen red.

    Examples 6 to 10 illustrate the need for what we have chosen to call *scripting*, using events or conditions in one media item to control the display of a second. Examples 6 and 7 refer to *a particular* instance of such conditions, the first occurrence in both these cases. Examples 8, 9 and 10 refer to *all* instances of such conditions. Note that these conditions that may hold at a specific time point (staccato) or over an interval (crescendo). Example 9 is a somewhat different example of scripting, where the operation to be performed is a housekeeping task, counting the number of chords, rather than the display of a media item.

This definition of the Beethoven problem illustrates each of the eight items listed under functional requirements. It also illustrates some, but not all, of the general requirements appearing in Section 2.1. Specifically, the notions of parallel and sequential composition are illustrated, but the notions of variable speed display and of reader intervention are not included. In order to illustrate these latter points, we now present two additions to the Beethoven problem, though these additions will not form part of the formal *Mexitl* specification to be presented in Section 5.

1. The reader may interrupt and stop the audio occurring at the beginning of the presentation (introduced in Part 2 above) and proceed directly to the start of the first movement.
2. The reader may stop and rewind the tape containing any movement. The reader may then replay that movement, and at any subsequent point may fast-forward to the end of that movement.

3 Introduction to *Mexitl*

We present a *core language* for *Mexitl*, which contains the primitive constructs of the notation. Then we describe the model theory underlying the language; this theory is based upon finite sequences of states (called *intervals*) and we define the satisfaction relation, which characterises the intervals that satisfy a logical formula.

3.1 The Core Language

Expressions The core language uses a simple expression notation. Expressions have the following form:

$$E \ ::= \ c \mid v \mid V \mid f(E) \mid \mathbf{mylen}$$

where $c \in \mathcal{N}$, $v \in \mathbf{Var}_{static}$, the set of static variables, $V \in \mathbf{Var}_{state}$, the set of state variables, and f is in a set of assumed functions (in some areas of the literature static and state variables are called respectively rigid and flexible variables [15]). We also assume a set $\mathbf{Var} = \mathbf{Var}_{static} \cup \mathbf{Var}_{state}$ of variables.

In addition, **mylen** is a distinguished variable which denotes the length of the current interval. This operator reflects the finiteness of models for *Mexitl*. The inclusion of **mylen** is a departure from standard interval temporal logic. We discuss the motivation for its inclusion in the next subsection.

The Logic The domain of logical propositions is denoted $\mathcal{P}$ and $P \in \mathcal{P}$ is constructed as follows:

$$\begin{aligned} P ::= &\ \mathbf{read}_X(V) \mid a_X(E) \mid a_X \mid p(E_1, ..., E_n) \mid E = E \mid \mathbf{False} \mid P \Rightarrow P \mid \\ &\ P \mathbin{;} P \mid P \ \mathbf{proj}\ P \mid P \ \tilde{;}\ P \mid (\exists x \leq E)P \mid P \ \mathbf{filter}\ P \end{aligned}$$

Among the above propositions, **read** and a belong to the set **Act** of actions and X is a set of such actions. Also p is in a set of given predicates; V is a state variable and E is an expression, as defined above.

Much of this logic will be well known to a reader familiar with interval temporal logic.

- **False** and $P \Rightarrow P$ are the familiar connectives of classical propositional logic.
- $p(E_1, ..., E_n)$ denotes application of a predicate to n expressions. In standard fashion, we will often write binary predicates infix.
- $E = E$ gives equality of expressions.
- **;** is the sequencing operator, *chop*, familiar from [16]. An interval satisfies $P \mathbin{;} Q$ if the interval can be divided into two contiguous sub-intervals, such that P holds over the first subinterval and Q holds over the second.
- **proj** is the projection operator, also described in [16]. An interval satisfies P **proj** Q if it can be sub-divided into a series of sub-intervals each of which satisfies P - we call P *the projection formula* - and the new interval formed from the end points of these sub-intervals satisfies Q, which we call *the projected formula*.

The reader who requires a more detailed discussion of these operators is referred to [16]. The remainder of our operators are not standard, and thus, require a little more explanation.

Actions. Actions in *Mexitl* are atomic, in the sense that they cannot be analysed into simpler components. Time is discrete, and an action is thought of as taking place in a single state; composite actions are built as *Mexitl* combinations of atomic actions using operators such as **;**.

read(V), $a(E)$ and a define the forms that actions can take. **read**(V) is a distinguished action; it is the only input action. **read**(V) enables new values to be bound to the variable V in the current state. User-defined actions, over which a ranges in the syntax above, will be defined in the separate abstract data typing notation, with their associated type information. Simple non-parametrised actions can be specified by referencing an action, a say, without any data parameter. Thus, this action can occur at a state without accessing the bindings at that state. In contrast, when actions of the form $a(E)$ occur at a state, the value of E with respect to the bindings at that state is associated with the occurrence. One example of such an action is the output of the value of an expression.

From a logical point of view we can think of an action a as an atomic proposition and of $a(E)$ as an application of an atomic predicate.

An action can appear a number of times in an interval; however, each of these represents a different instance of the action. In particular, action occurrences cannot span two states in an interval. Thus, at the level of interval states, actions do not have duration. However, durational behaviour can be obtained by defining composite actions, which are a shorthand for the occurrence of multiple primitive actions. In particular, primitive actions may correspond to indexing into a composite action. For example, an action $playFrame[500]$ might be a constituent of the composite action $playVideo$.

Although actions do not have duration, sets of (distinct) actions can occur at the same state. Such sets reflect simultaneous lock-step occurrence of the actions. In this sense, the model employs synchronous parallelism.

Framing of actions. One aspect which distinguishes our usual perception of logical propositions and actions is the idea of framing. An assertion of a, where a is a particular action, is often interpreted as '*a and no other action* happens' whereas a logical interpretation simply reads this as a happening. The former interpretation, in which the action a is "framed", would lead to a non-monotonic logic were we to adopt it.

Instead of this in our system we subscript the actions with sets X of actions. a_X is interpreted as 'a happens and none of the other actions in X happens'. The set X thus provides an explicit frame within which the action a takes place. Logically the interpretation of a_X is the conjunction of a and $\neg b$ for all b in $X - \{a\}$.

We add a distinguished action – **null** – to the set of actions. This action has null effect, but can be used for framing purposes thus: $\mathbf{null}_X$.

The reason for adding this local framing as primitive is that it supports the composition of separate channels in a modular way as will be seen in the examples below.

A restriction of our model is that it does not support *auto-concurrency*, i.e. multiple instances of the same action occurring at the same state. This slightly restricts generality, however the extra expressiveness obtained does not seem to be needed in our particular area of application.

Length and Next. In contrast to the standard approach to ITL we have not included the next operator, $\bigcirc$, directly in *Mexitl*. However, standard length operators and $\bigcirc$ can be derived from the expression **mylen**, as follows:

$$\mathbf{lesseq}(E) \equiv \mathbf{mylen} \leq E$$

$$\mathbf{len}(E) \equiv \mathbf{mylen} = E$$

$$\bigcirc P \equiv (\mathbf{mylen} = 1) \; ; \; P$$

Thus, only intervals whose length is less than or equal to the value of E will satisfy $\mathbf{lesseq}(E)$, while only intervals of length E will satisfy $\mathbf{len}(E)$. The latter of