Modelling Linguistic Events

Miguel Leith and Jim Cunningham

Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London, England

E-mails: *mfl@doc.ic.ac.uk* and *rjc@doc.ic.ac.uk*

Abstract. Linguistic categories such as progressives, perfectives and tense are at the heart of our ability to describe events in natural language. We show how temporal readings of simple sentences involving these categories may be expressed concisely using a fragment of Halpern and Shoham's interval logic HS, and describe how to obtain the readings using a compositional semantic approach. By modelling the readings as timelines we show how linguistic entailments may be checked simply and systematically.

Category: Tense and Aspect in Natural Language.

1 Introduction

Linguistic categories such as progressives, perfectives and tense are at the heart of our ability to describe events in natural language. Through these categories, which we will loosely call *temporal*, we express the relative positions and compositions of events in time, characterised by temporal relations such as *later than, adjoins, overlaps, begins, ends* or *includes*. For example the simple sentence *Max stopped running* might give rise to the informal temporal reading in Fig.1.

Even for quite simple sentences it is hard to find a straightforward formalism for representing temporal readings that may be arrived at by a compositional semantic analysis and that may be modelled intuitively so that entailments may be checked easily. Our work here focuses on a formalism which meets these criteria for a range of simple examples.

The logical representation of tense and aspect in natural language has considerable tradition and many open issues which will not be discussed here. The reader is referred to the recent book by Verkuyl [10]. We take as our point of departure a thesis by Kent [4]. Kent draws on work of Moens [7], Lascarides [5], and Moens and Steedman [8] in order to develop a formal account of linguistic behaviour in his own interval logic. His account provides evidence for the adequacy of interval models in explicating the entailments between simple related sentences.

We too have adopted a formal semantic approach and chosen an interval logic to represent temporal readings. We analyse sentences using compositional semantic rules into readings expressed using the interval tense logic known as

Fig. 1. Time periods for the sentence *Max stopped running.*

HS, which was described by Halpern and Shoham in [2]. We define a fragment of HS which is sufficient for our expressive needs and for which formulae may be modelled as intuitive data structures called timelines. Pairs of timelines may then be compared as a way of checking linguistic entailments.

The structure of this paper, then, is as follows. In Sect.2 we give a summary of the logic HS, and discuss the modal operators which are of immediate use to us. In Sect.3 we describe our logic fragment HSF1 and our method for constructing timeline models. We then outline an interesting type of timeline that we term a Closest Position Model. In Sect.4 we describe a systematic way of obtaining the HSF1 temporal readings of simple sentences using a compositional semantic approach. This leads us to give examples of the analysis of sentences into their temporal readings and thence into timelines. Examples of the types of sentences we will be looking at are given in Fig.2.

1 - *Mike is playing tennis.*
2 - *Max stopped running.*
3 - *Jack has finished eating dinner.*
4 - *Mark will start racing.*
5 - *Fred has been playing the piano.*

Fig. 2. Example sentences for which we wish to model temporal readings.

In Sect.5 we summarise the details of an entailment checker which can carry out comparisons between timelines, and demonstrate its use in modelling linguistic entailments. Examples of the types of entailments we wish to check are given in Fig.3.

1 - *Max stopped running* $\Rightarrow$ *Max was running*
2 - *Jack stopped eating dinner* $\not\Rightarrow$ *Jack finished eating dinner*
3 - *Mike is playing tennis* $\Rightarrow$ *Mike has been playing tennis*

Fig. 3. Example entailments to check.

Finally in Sect.6 we make our conclusions.

2 The Logic HS

HS is the interval tense logic of Halpern and Shoham, which was first described in [2] and further investigated by Venema in [9]. In HS the set of possible worlds is the set of closed intervals in a temporal frame which may be dense or discrete. Its six basic modal operators and six common derived modal operators correspond to relative interval positions as illustrated by the diagram in Fig.4. We give additionally the derived beginning and end point operators. The corresponding necessitation operators are defined in the usual way.

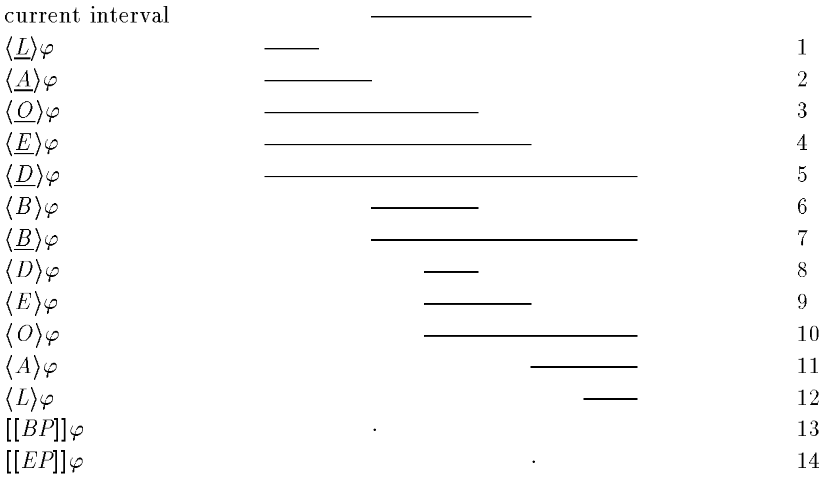

Fig. 4. Pictorial illustration of relative interval positions and their HS formulae

One may notice a similarity between some of the HS operators and those of Allen's interval logic, described in [1]. Specifically, DURING($i1$, $i2$) is similar to $\langle D \rangle$ and $\langle \underline{D} \rangle$, STARTS($i1$, $i2$) to $\langle B \rangle$ and $\langle \underline{B} \rangle$, FINISHES($i1$, $i2$) to $\langle E \rangle$ and $\langle \underline{E} \rangle$, BEFORE($i1$, $i2$) to $\langle L \rangle$ and $\langle \underline{L} \rangle$, OVERLAP($i1$, $i2$) to $\langle O \rangle$ and $\langle \underline{O} \rangle$, MEETS($i1$, $i2$) to $\langle A \rangle$ and $\langle \underline{A} \rangle$.

2.1 A Subset of HS Operators

An interesting subset of HS operators is the union of the set of all possibility operators with one of the necessitation operators, $[D]$, which means informally *all intervals during the current interval.* The $[D]$ operator is of particular use here as it may be used to effect *homogeneity*, an important concept in event representation. A proposition is said to be homogeneous if it is true at all sub-intervals of any intervals at which it holds. Homogeneity is used by Kent in [4] as a property on states. If someone was running for five minutes, for instance, then they were running at all periods of time within that five minutes, unless an exception is specified, in which case the proposition may be split into two or more homogeneous periods instead. By having a formula $[D]$*(run(max))* hold at an interval we would be denoting that Max is running at all intervals/points during that interval.

3 HSF1: A Fragment of HS Logic

We begin this section by defining the fragment HSF1 which we later use to represent temporal readings of sentences. We describe how to model the formulae from the fragment as *timelines*. Lastly we give a full example of a timeline being constructed, and define a type of timeline that is of particular interest.

The syntax of HS is defined using grammar rules as follows.

HSF1 : Exp2 | PossOp HSF1
PossOp : $\langle D\rangle \mid \langle \underline{D}\rangle \mid \langle E\rangle \mid \langle \underline{E}\rangle \mid \langle B\rangle \mid \langle \underline{B}\rangle \mid \langle L\rangle \mid \langle \underline{L}\rangle \mid$
$\langle A\rangle \mid \langle \underline{A}\rangle \mid \langle O\rangle \mid \langle \underline{O}\rangle \mid [[EP]] \mid [[BP]]$
Exp2 : ExtendedAtomic | HSF1 $\wedge$ HSF1
ExtendedAtomic : Atomic | $[D]$ Atomic
Atomic : PropConstant | $\neg$ PropConstant
PropConstant : $p_1 \mid p_2 \mid p_3 \ldots$

We impose a restriction on frames by requiring that the set of points T in a temporal frame is dense, linear and unbounded.

3.1 Modelling HSF1 Formulae

Proposition 1. *Any HSF1 formula may be modelled by a finite number of alternative data structures which we will refer to as timelines.*

Definition 2. A timeline is a triple $\langle$*Start Interval*, *Ordering*, *Interval Mapping*$\rangle$ where *Start Interval* is **now**, *Ordering* is a sequence of point labels in linear dense unbounded time and *Interval Mapping* is a mapping between intervals and the sets of extended atomic formulae that hold at them. An interval is represented by a pair of point labels from the ordering e.g. $(Pl_1,\ Pl_2)$.

Proposition 1 is justified because in the case of the HSF1 fragment, the semantics of an HSF1 formula can be interpreted by a procedure which builds the timelines that may model it. The procedure by which timelines are built is defined informally below. In general there may be more than one possible timeline for a given HSF1 formula although some formulae may give rise to only one.

Timelines are built by an algorithm called the Timeline Construction Algorithm (TCA). The algorithm is a recursive traversal of an HSF1 formula that builds an ordering and an interval mapping. At each stage of the recursive traversal we are at a particular *current interval* denoted by a pair of point labels e.g. $(Pl_1,\ Pl_2)$. Each modal operator (except for $[D]$) is a possibility operator in its positive form (e.g. $\langle E\rangle$, $\langle L\rangle$, $[[EP]]$ etc.) and effectively asserts the existence of an interval in given temporal relation to the current interval. Thus the effect of encountering a possibility operator in the algorithm is to change the current interval and update the ordering to reflect the points of the new current interval in their correct positions. If the algorithm encounters an extended atomic formula it will augment the interval mapping to reflect the formula being true at the current interval. For instance if it encounters a formula $[D](p)$ (or a formula p) at current interval $(Pl_4,\ Pl_3)$ it will add $[D](p)$ (or p) to the set of formulae holding at $(Pl_4,\ Pl_3)$.[1] We henceforth refer to a single entry of an interval mapping $((Pl_x,\ Pl_y) : \{\ F_1,\ F_2 \ldots F_n\ \})$ as a *binding*.

By the preceding argument a timeline modelling an HSF1 formula of depth n (where we count the starting interval and each possibility modality as adding 1 to the depth) has at most $2n$ time points in its ordering. When we take into account the possible branching effects when new intervals are introduced, a potentially large but finite set of alternative timelines may arise.

3.2 Timeline Construction Example

The TCA for an example HSF1 formula $\langle \underline{L}\rangle(x \wedge \langle B\rangle(y \wedge \langle A\rangle(z)))$ proceeds as described in the example over Figs. 5 to 8. Note that the point labels of timelines are just that, labels, their numbering is arbitrary and does not correspond to points on a number line.

As the TCA proceeds the ordering and interval mapping evolve, while the start interval remains constant. The ordering is modified whenever a possibility operator is analysed and the interval mapping is modified when an extended atomic formula is analysed. Fig.5 shows the initial state of the algorithm. In subsequent figures the ordering and interval mapping are shown only at states where they have just been modified.

In Fig.6, state 2 shows the situation just after analysis of the $\langle \underline{L}\rangle$ operator. Note that a line over a point label means it was a label in the previous current interval and a line under a point label means that it is a label in the new current interval. Following this we come to a conjunction. The left formula of the conjunction, x, is analysed first (Fig.6, state 3), then the right formula (Fig.6, state 4). We then come to the lower conjunction and the left formula is analysed first, namely y (Fig.6, state 5).

Next, the right formula of the lower conjunction is considered. Its top operator, $\langle A\rangle$, is analysed. This leads to alternative orderings due to choices when determining the position of the new current interval (see Fig.7). Finally we come to the deepest sub-formula, namely z (see Fig.8), where the algorithm terminates.

---

[1] By the semantics of the operator $[D]$, the formula $[D](p)$ must hold at all internal durative intervals and p must hold at all internal intervals/points. These possible inferences must be reflected in any algorithms that reason using timelines.

Current interval is (Pl_1, Pl_2). *Next formula is* $\langle \underline{L} \rangle (x \wedge \langle B \rangle (y \wedge \langle A \rangle (z)))$.

Start interval: (Pl_1, Pl_2). Ordering: $[Pl_1, Pl_2]$. Interval Mapping: $\langle\rangle$.

Fig. 5. Timeline construction example, state 1

Operator just analysed is $\langle \underline{L} \rangle$. *Next formula is* $x \wedge \langle B \rangle (y \wedge \langle A \rangle (z)))$.

Ordering: $[\underline{Pl_4}, \underline{Pl_3}, \overline{Pl_1}, \overline{Pl_2}]$.

Formula just analysed is x. *Current interval is* (Pl_4, Pl_3).
Next formula is $\langle B \rangle (y \wedge \langle A \rangle (z)))$.

Interval Mapping: $\langle (Pl_4, Pl_3)\colon \{\ x\ \})\ \rangle$

Operator just analysed is $\langle B \rangle$. *Next formula is* $y \wedge \langle A \rangle (z)$.

Ordering: $[\underline{\overline{Pl_4}}, \underline{Pl_5}, \overline{Pl_3}, Pl_1, Pl_2]$.

Formula just analysed is y. *Current interval is* (Pl_4, Pl_5). *Next formula is* $\langle A \rangle (z)$.

Interval Mapping: $\langle (Pl_4, Pl_3)\colon \{\ x\ \}, (Pl_4, Pl_5)\colon \{\ y\ \}\ \rangle$.

Fig. 6. Timeline construction example, states 2 - 5 (top to bottom)

Operator just analysed is $\langle A \rangle$.

First alternative:
New current interval is (Pl_5, Pl_6).
Ordering: $[\overline{Pl_4}, \underline{\overline{Pl_5}}, \underline{Pl_6}, Pl_3, Pl_1, Pl_2]$.

Second alternative:
New current interval is (Pl_5, Pl_3).
Ordering: $[\overline{Pl_4}, \underline{\overline{Pl_5}}, \underline{Pl_3}, Pl_1, Pl_2]$.

Third alternative:
New current interval is (Pl_5, Pl_6).
Ordering: $[\overline{Pl_4}, \underline{\overline{Pl_5}}, Pl_3, \underline{Pl_6}, Pl_1, Pl_2]$.

Fourth alternative: ...

Next formula is z.

Fig. 7. Timeline construction example, state 6

Formula just analysed is z.

First alternative:

Interval Mapping: $\langle (Pl_4, Pl_3)$: { x }, (Pl_4, Pl_5): { y }, (Pl_5, Pl_6): { z } $\rangle$

Second alternative:

Interval Mapping: $\langle (Pl_4, Pl_3)$: { x }, (Pl_4, Pl_5): { y }, (Pl_5, Pl_3): { z } $\rangle$

Third alternative:

Interval Mapping: $\langle (Pl_4, Pl_3)$: { x }, (Pl_4, Pl_5): { y }, (Pl_5, Pl_6): { z } $\rangle$

Fourth alternative: ...

Algorithm finished.

Fig. 8. Timeline construction example, state 7

3.3 Closest Position Models

Of the timelines that may model an HSF1 formula, there will be one timeline that we term its *closest position model* (CPM). The idea arises as follows. Each time the TCA interprets the semantics of a possibility operator such as $\langle \underline{A} \rangle$, $\langle \underline{E} \rangle$ etc. it changes the current interval by selecting point labels in appropriate positions in the timeline ordering (see Figs. 6 and 7). Whenever a new point is to be inserted before/after some point label Pl_n the CPM is constructed by selecting a new point label immediately before/after Pl_n in the ordering.

For example the first alternative in the analysis of the operator $\langle A \rangle$ in Fig.7 enables a CPM. (In Fig.6 states 2 and 4 there are no alternatives).

CPMs separate future, past, and present, in preference to alternatives with overlapping intervals. They are required for the simple compositional semantics given in Sect.4. Natural language entailments can be demonstrated without restricting the class of models (see [6]), but psychological interpretation may not consider all logical alternatives (see [3] for related discussion).

4 Temporal Sentence Semantics

In this section we show how formulae from the HSF1 fragment may be used to represent the temporal readings of the sentences given in Fig.2.

4.1 The Treatment of Start, Stop and Culmination

In our temporal readings of natural language sentences great importance is given to three special types of time point: a state's start point, stop point and culmination point. In this subsection we define each of the three point types in terms of the HSF1 formulae that hold at them and illustrate the principles with diagrams. But first we define what we mean by a state.

Definition 3. An HSF1 state is a formula of the form $[D](P)$ where P may be any atomic formula. The state of a particular atomic proposition p is on at any interval/point at which $\langle \underline{D} \rangle([D](p))$ holds. The state of a particular atomic proposition p is off at any interval/point at which $\langle \underline{D} \rangle([D](\neg p))$ holds.

A start point of a state is a point of transition from the state being off to the state being on. This means that $[D](\neg p)$ must hold at an interval adjoining the point on the left, and $[D](p)$ must hold at an interval adjoining the point on the right.

Definition 4. A start point of a state $[D](p)$ is any point at which the formula $\langle \underline{A} \rangle([D](\neg p)) \wedge \langle A \rangle([D](p))$ holds.

| a | b |

Index
a - $[D](\neg p)$
b - $[D](p)$

Fig. 9. Start point of a state

A stop point of a state is a point of transition from the state being on to the state being off. This means that $[D](p)$ must hold at an interval adjoining the stop point on the left, and $[D](\neg p)$ must hold at an interval adjoining the stop point on the right.

Definition 5. A stop point of a state $[D](p)$ is any point at which the formula $\langle \underline{A} \rangle([D](p)) \wedge \langle A \rangle([D](\neg p))$ holds.

| a | b |

Index
a - $[D](p)$
b - $[D](\neg p)$

Fig. 10. Stop point of a state

A culmination point of a state is a point at which the state is said to be *finished* or *completed*. Culmination points are critical to our understanding of linguistic expressions such as *Jill finished eating dinner*. An issue for some time in event representation has been the question of what a proposition's truth value is at a stop or start point of a state. We define a culmination of a state as a stop point of the state for which the state's atomic proposition is true. Formally, this is defined below.

Definition 6. A culmination point of a state $[D](p)$ is any stop point of $[D](p)$ at which p holds.

| a $|_b$ c |

Index
a - $[D](p)$
b - p
c - $[D](\neg p)$

Fig. 11. Culmination point of a state

4.2 Obtaining Temporal Readings Compositionally

To work out the HSF1 temporal readings of simple sentences like those in Fig.2 we adopt a compositional approach. Semantic operators correspond to syntactic features and are represented as lambda terms which are combined in a prescribed order. The composite lambda term is then reduced to yield the temporal reading of the sentence. We describe the lambda terms next.

PERF: λw.$\langle \underline{A} \rangle$(w)

Feature correspondence(s): auxiliary *have* coupled with perfective form of verb.

PAST: λx.$\langle \underline{L} \rangle$(x)

Feature correspondence(s): past tense form of main verb or auxiliary.

PRES: λx.x

Feature correspondence(s): present tense form of main verb or auxiliary.

FUTURE: λx.$\langle L \rangle$(x)

Feature correspondence(s): auxiliary *will* coupled with base form of verb.

BE: λy.λf.$\langle \underline{D} \rangle$((f)(y))

Feature correspondence(s): auxiliary *be*.

CULM: λy.λf.$[[EP]]$(y $\wedge$ $\langle \underline{A} \rangle$((f)(y)) $\wedge$ $\langle A \rangle$((f)($\neg$y)))

Feature correspondence(s): a verb such as *finish* that records a sense of accomplishment in a verb phrase e.g. *finish reading the book*.

START: λy.λf.$[[EP]]$(($\langle \underline{A} \rangle$((f)($\neg$y) $\wedge$ $\langle A \rangle$((f)(y)))

Feature correspondence(s): a verb denoting the starting of an activity e.g. *start*.

STOP: λy.λf.$[[EP]]$($\langle \underline{A} \rangle$((f)(y)) $\wedge$ $\langle A \rangle$((f)($\neg$y)))

Feature correspondences: a verb denoting the stopping of an activity e.g. *stop*.

ING: $\lambda z.[D](z)$
Feature correspondence(s): a *ing*-inflection of a main verb denoting a process of some kind, for instance *running*, *eating*.

PROP: p (A lambda calculus constant, some atomic proposition for a simple sentence with the verb in its base form, e.g. *run(max)*, *eat(jill, apple)*).
Feature correspondence(s): the basic temporally unmodified meaning of a sentence e.g. *Max run*, *Jill eat apple*.

The lambda terms are combined into a larger term in the following structure:

(Pos1)(Pos2)(((Pos3)PROP)ING)

where Pos1 may be occupied by PERF, Pos2 may be occupied by PAST, PRES or FUTURE and Pos3 may be occupied by BE, CULM, STOP or START. This term may then be reduced to obtain the temporal reading.

4.3 Example Temporal Readings

We give here our temporal readings of the sentences listed in Fig.2.

<u>Sentence Semantics</u>
(PRES)(((BE)(*play(mike, tennis)*))(ING)) $\equiv$
$(\lambda x.x)(((\lambda y.\lambda f.\langle \underline{D}\rangle((f)(y)))(play(mike,\ tennis)))(\lambda z.[D](z)))$
reduces to
$\langle \underline{D}\rangle([D](play(mike,\ tennis)))$

<u>Closest Position Model</u>

| a |

| * |

<u>Timeline Index</u>
a: $[D]$*(play(mike, tennis))* *: *now*

Fig. 12. A temporal reading of the sentence *Mike is playing tennis*

The present progressive results in a timeline like the one in Fig.12.

Sentence Semantics

$(\mathrm{PAST})(((\mathrm{STOP})(run(max)))(\mathrm{ING})) \equiv$

$(\lambda\mathrm{x}.\langle \underline{L} \rangle(\mathrm{x}))(((\lambda\mathrm{y}.\lambda\mathrm{f}.[[EP]](\langle \underline{A} \rangle((\mathrm{f})(\mathrm{y})) \wedge \langle A \rangle((\mathrm{f})(\neg\mathrm{y}))))(run(max)))(\lambda\mathrm{z}.[D](\mathrm{z})))$

reduces to

$\langle \underline{L} \rangle([[EP]](\langle \underline{A} \rangle([D](run(max))) \wedge \langle A \rangle([D](\neg run(max)))))$

Closest Position Model

| b | a | | * |

Timeline Index

b: $[D](run(max))$ *a*: $[D](\neg run(max))$ *: *now*

Fig. 13. A temporal reading of the sentence *Max stopped running*

The stopped event in Fig.13 is placed in the past.

Sentence Semantics

$(\mathrm{PERF})(((\mathrm{CULM})(eat(jack,\ dinner)))(\mathrm{ING})) \equiv$

$(\lambda\mathrm{w}.\langle \underline{A} \rangle(\mathrm{w}))(((\lambda\mathrm{y}.\lambda\mathrm{f}.[[EP]](\mathrm{y} \wedge \langle \underline{A} \rangle((\mathrm{f})(\mathrm{y})) \wedge \langle A \rangle((\mathrm{f})(\neg\mathrm{y}))))(eat(j,\ d))))(\lambda\mathrm{z}.[D](\mathrm{z})))$

reduces to

$\langle \underline{A} \rangle([[EP]](eat(jack,\ dinner) \wedge \langle \underline{A} \rangle([D](eat(jack,\ dinner))) \wedge \langle A \rangle([D](\neg eat(jack,\ dinner)))))$

Closest Position Model

| c |$_a$ * |

| b |

Timeline Index

c: $[D](eat(jack,\ dinner))$ *b*: $[D](\neg eat(jack,\ dinner))$

a: $eat(jack,\ dinner)$ *: *now*

Fig. 14. A temporal reading of the sentence *Jack has finished eating dinner*

Perfectives convey a sense of current significance or recency. For instance *Jack has finished eating dinner* appears to denote that the completion of the event took place at a time that is in touch with the current temporal focus i.e. **now**. We attempt to capture this sense by applying the operator $\langle \underline{A} \rangle$ to represent the perfective, meaning that it adjoins the beginning of **now**. The right point of **now** may be seen as the current instant in time.

Sentence Semantics

(FUTURE)(((START)(*race(mark)*))(ING)) $\equiv$

$(\lambda x.\langle L\rangle(x))(((\lambda y.\lambda f.[[EP]](\langle \underline{A}\rangle((f)(\neg y) \wedge \langle A\rangle((f)(y))))(race(mark)))(\lambda z.[D](z)))$

reduces to

$\langle L\rangle([[EP]](\langle \underline{A}\rangle([D](\neg race(mark))) \wedge \langle A\rangle([D](race(mark)))))$

Closest Position Model

$| \quad * \quad | \qquad | \quad b \quad | \quad a \quad |$

Timeline Index

b: $[D](\neg race(mark))$ *a*: $[D](race(mark))$ *: *now*

Fig. 15. A temporal reading of the sentence *Mark will start racing*

The start of the racing event in Fig.15 is placed in the future.

Sentence Semantics

(PERF)(PRES)(((BE)(*play(fred, piano)*))(ING)) $\equiv$

$(\lambda w.\langle \underline{A}\rangle(w))(\lambda x.x)(((\lambda y.\lambda f.\langle \underline{D}\rangle((f)(y)))(play(fred,\ piano)))(\lambda z.[D](z)))$

reduces to

$\langle \underline{A}\rangle(\langle \underline{D}\rangle([D](play(fred,\ piano))))$

Closest Position Model

$| \qquad a \qquad |$

$\qquad\quad | \qquad * \qquad |$

Timeline Index

a: $[D](play(fred,\ piano))$ *: *now*

Fig. 16. A temporal reading of the sentence *Fred has been playing the piano*

The perfective in Fig.16 does not have the sense of completion that was present in Fig.14 since there is no culmination, but the *playing piano* state runs into **now** as an ongoing event.

5 An Entailment Checker for HSF1 Formulae

In the general case an entailment between two HSF1 formulae holds if each timeline modelling the left formula entails a timeline modelling the right formula.

Therefore if we determine a single timeline for each HSF1 formula there is a one-to-one correspondence between an HSF1 entailment and its timeline entailment. An entailment between a pair of timelines holds if the timeline of the right formula is a *substructure* of the timeline of the left formula, which in informal terms means it will *fit* into it. We refer to the algorithm which compares the timelines as the Substructure Fitting Algorithm (SFA). The algorithm succeeds if a new timeline may be created which represents the match of the two timelines. We will use the terms *substructure timeline*, *superstructure timeline* and *matched timeline* to distinguish between the operative timelines in this algorithm.

The objective of the SFA is to build a new timeline (the matched timeline) in which every formula in every binding of the substructure timeline is matched by a formula from a binding in the superstructure timeline in one of the ways depicted in Fig.17.

At each stage of the recursive algorithm, a binding is removed from the substructure interval mapping and an attempt is made to match its formulae such that the matched ordering that the algorithm is building remains consistent with the superstructure ordering and substructure ordering. This is repeated until either the substructure interval mapping has been emptied (success) or it cannot be emptied (failure). Full details of the SFA will be given in a forthcoming thesis.

Exact match, atomic case	Exact match, homogeneous case
[*prop*]	[$[D](prop)$]
[*prop*]	[$[D](prop)$]
Right homogeneous match	**Left homogeneous match**
[$[D](prop)$]	[$[D](prop)$]
[$[D](prop)$]	[$[D](prop)$]
Internal match, homogeneous case	**Internal match, atomic case**
[$[D](prop)$]	[$[D](prop)$]
[$[D](prop)$]	[*prop*]

Fig. 17. How pairs of formulae may be matched by the SFA

5.1 Entailment Examples

Having summarised the principles of the operation of our entailment checker, we can now move on to give some examples of its results. In Figs. 18 and 19 we show checks for the logical entailments corresponding to two of the natural language entailments in Fig.3.

Entailment Check
$\langle \underline{L} \rangle([[EP]](\langle A \rangle([D](\neg\ run(max))) \wedge \langle \underline{A} \rangle([D](run(max))))) \models_{cpm}$
$\langle \underline{L} \rangle(\langle \underline{D} \rangle([D](run(max))))$

Superstructure CPM

B	A		*

Timeline Index
B: $[D](run(max))$ A: $[D](\neg run(max))$
*: *now*

Substructure CPM

a		*

Timeline Index
a: $[D](run(max))$ *: *now*

Match Timeline

Ba	A		*

Fig. 18. Entailment check for: *Max stopped running* $\Rightarrow$ *Max was running*

6 Conclusion

In this paper we have demonstrated a route to compositional temporal semantics of natural language events using interval temporal logic. The straightforwardness of the semantic rules suggests to us that certain HS operators may be close in function to uses of the temporal linguistic categories that were analysed. We believe that the timeline models and the treatment of culmination and the perfective are a contribution towards intuitive formal semantics of natural language events. The restrictions on the fragment of the logic and the models we allow in order to achieve this may be of psychological relevance and a hint for a new logic.

References

1. James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.
2. Joseph Halpern and Yoav Shoham. A propositional modal logic of time intervals. In *Proceedings of Symposium on Logic in Computer Science*, pages 279–292, Cambridge, Massachusetts, June 16-18, 1986.

Entailment check

$\langle \underline{D} \rangle ([D](play(mike, tennis))) \models_{cpm} \langle \underline{A} \rangle (\langle \underline{D} \rangle ([D](play(mike, tennis))))$

Superstructure CPM

```
            |         A         |
                  |  *  |
```

Timeline Index

A: $[D](play(mike, tennis))$ $*$: *now*

Substructure CPM

```
            |     a     |
                  |     *     |
```

Timeline Index

a: $[D](play(mike, tennis))$ $*$: *now*

Match Timeline

```
      |            A            |
      |     a     |
            |     *     |
```

Fig. 19. Entailment check for: *Mike is playing tennis* $\Rightarrow$ *Mike has been playing tennis*

3. P.N. Johnson-Laird and Ruth M.J. Byrne. *Deduction.* Erlbaum, 1991.
4. Stuart Kent. *Modelling Events from Natural Language.* PhD thesis, Imperial College, London, 1993.
5. Alex Lascarides. *A Formal Semantic Theory of the Progressive.* PhD thesis, University of Edinburgh, 1988.
6. Miguel Leith and Jim Cunningham. Representing and reasoning with events from natural language. In *First International Joint Conference on Qualitative and Quantitative Practical Reasoning*, 1997.
7. Marc Moens. *Tense, Aspect and Temporal Reference.* PhD thesis, University of Edinburgh, 1987.
8. Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics 14*, pages 15–28, 1988.
9. Yde Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.
10. Henk Verkuyl. *A theory of aspectuality.* Cambridge University Press, 1993.

This article was processed using the LaTeX macro package with LLNCS style