Improving String Processing for Temporal Relations

David Woods, Tim Fernando
ADAPT Centre
Computational Linguistics Group
School of Computer Science and Statistics
Trinity College Dublin, Ireland
dwoods@tcd.ie, tim.fernando@tcd.ie

Abstract

This paper presents a refinement of the superposition operation on strings which are used to represent temporal relation information such as is found in documents annotated with TimeML. Superposition is made demonstrably more efficient by interleaving generation with testing, rather than generating and then testing. The strings offer compact visual appeal while remaining an attractive option for computation and reasoning. Motivated by Freksa's semi-interval relations, a suggestion is also made for a potential method of representing partial information in these strings so as to allow for analysis at different granularities, and for more flexibility when dealing with cases of ambiguity.

1 Introduction

A string of n sets $\alpha_1\alpha_2 \cdots \alpha_n$ can be used to represent a sequence of events and time periods such that the linear order and inter-relations of the events are clearly apparent, while remaining compact. Such a string is read from left to right chronologically, with each α_i, $i \in \{1, 2, \ldots, n\}$, depicting one of n moments in time, and specifying the set of exactly those temporal propositions, or fluents, which hold simultaneously at that moment i. A fluent $a \in \alpha_i$ is understood to be occurring before another fluent $a' \in \alpha_j$ iff $i < j$ and $a' \notin \alpha_i$.

Note that these strings do not (necessarily) offer any information concerning real duration: a fluent may occur in several string positions, but this does not affect any interpretation of its duration, only its relation to other fluents, i.e. if the symbol a appears in both α_i and α_{i+1}, the event it stands for is not understood as being twice as long as if the symbol had only appeared in α_i. Fluents representing fixed time points may be used to give a sense of real time (e.g. $a =$ "5pm on 25th May 2018"). Because of this, a string in which $\alpha_i = \alpha_{i+1}$ for any $1 \leq i < n$ will not have its interpretation affected if either α_i or α_{i+1} are deleted. For example, the string $\{a\}\{a\}\{a,b\}\{b\}\{b\}$ is equivalent in interpretation to $\{a\}\{a,b\}\{b\}$. A string featuring these repetitions is said to *stutter*, and the process of removing stutter from a string is called *block compression* (Fernando, 2015; Woods et al., 2017).

Throughout this paper, each string position α_i will be drawn as a box, with $\boxed{}$ for the empty set $\varnothing$, allowing the strings to be read like strips of film. Events are treated as bounded intervals, such that they have a beginning and ending – this is represented through the use of bounding empty sets – although this assumption is not required by the string framework: the finite event a, drawn as $\boxed{}\boxed{a}\boxed{}$, as opposed to non-finite event a', drawn as $\boxed{a'}$.

By superposing multiple strings, large amounts of information may be condensed into a single string, which offers a timeline-like visual appeal. The strings may be used to encode and reason about interval relations, as in Allen (1983), and also to aid visualisation in the annotation process. It is hoped that this approach may be seen as complementary to existing graphical tools. In Section 3, a refinement is offered of the superposition operation discussed in Woods et al. (2017) which preserves relational information under reduct, and interleaves generation with testing of results for a more efficient calculation. A potential extension of the framework is also shown in Section 4 such as to allow for incomplete information, and for Freksa (1992)'s semi-interval relations to be represented. This would enable reasoning over partial or uncertain information, or in case coarser analysis was desired.

2 Motivation

Several attempts have been made to create a visualisation system which best represents the temporal relations found in TimeML (TimeML Working Group, 2005; Pustejovsky et al., 2010), particularly for use as an annotation aid, including Tango (Pustejovsky et al., 2003; Verhagen et al., 2006), using labelled arcs in a directed graph, and T-BOX (Verhagen, 2005a), using relative placement of boxed temporal propositions. A clear advantage of T-BOX is that it presents the information in a way that makes it easy to quickly gain a sense of a document's overall temporal structure, due to its intuitive use of arrows, box-inclusion, and stacking.

Using strings as a representational tool presents the same intuitive structure, with a timeline-like left-to-right layout, but in a compact format which may also be used for computations, rather than being a purely visual entity.

Although these strings may be used for other problems (see Fernando (2015)), one of their more obvious uses is as a tool for the simultaneous visualisation of and reasoning about collections of Allen (1983)'s interval relations, such as those which appear (slightly renamed) as TLINKs in documents annotated with TimeML. A TLINK, or temporal link, is a tag featuring a pair of temporal entities, either event instances or time periods, from the text of the document and a relation between them, which can be translated to a string.

By way of example, taking a tag `<TLINK lid="l5" relType="BEFORE" timeID="t86" relatedToTime="t82"/>`[1] ($t86$ referring to a five year period, and $t82$ being the document creation time), it is straightforward to convert this to a string: find the Allen relation which corresponds to the `relType` attribute, then substitute a and a' in Table 1 with the values of `timeID` (or `eventInstanceID`) and `relatedToTime` (or `relatedToEventInstance`), respectively, to give "$t86$ before $t82$" – $\boxed{\,}\boxed{t86}\boxed{\,}\boxed{t82}\boxed{\,}$, which is interpreted as saying that the five year period occurred before the document creation time. By converting all of a document's TLINKs to strings and superposing them with one another, a picture begins to build up of the document's overall temporal structure.

As Derczynski (2016, p. 2) points out, the choice of representation for temporal relation types is critical, as choosing the right relation for arbitrary pairs of events is difficult. By using these strings, an annotator would be able to see the state of the document as a whole, and receive suggestions when there are constraints on the possible relations they may choose, given other relations. For example, given "x during y" – $\boxed{\,}\boxed{y}\boxed{x,y}\boxed{y}\boxed{\,}$ and "y during z" – $\boxed{\,}\boxed{z}\boxed{y,z}\boxed{z}\boxed{\,}$, there is only one possible relation between x and z: "x during z" – $\boxed{\,}\boxed{z}\boxed{x,z}\boxed{z}\boxed{\,}$. In scenarios such as this, where there is only one possibility, the system should be able to fill it in automatically, but in other cases, it might suggest to an annotator that only a certain group of relations is feasible.

The ultimate goal here would be to achieve full temporal closure over the document. If the temporal entities were (as in Tango) represented as nodes on a graph, and the relations between them as the arcs, this would look like a fully connected graph, with an arc from every node to every other node. Using strings instead, a single, timeline-like string would be computable, which would contain all of the temporal relation information of the document. It would be possible to determine the relation between any two (or more) events in a single operation, and to quickly examine subsections of the timeline, or see the relative ordering of a specific subset of events (see reduct and projections, Section 3).

Reaching this somewhat lofty target in an automatic way seems unlikely, at least for now, as the number of relations given in a typical document from TimeBank (Pustejovsky et al., 2006), the largest corpus of documents annotated with TimeML, is far short of the total $N(N-1)/2$ (for N fluents) links mentioned by Verhagen (2005b, p. 3), and in most cases, there is simply not enough information to compute everything. See Section 4 for further discussion on this matter.

[1]Document ABC19980108.1830.0711.tml

One benefit the strings offer from an automation point of view is the ability to quickly determine inconsistent labellings in TimeML documents. If superposition of any two strings produces an empty language, then there is an incompatibility between them (e.g. "x before y" – $\begin{array}{|c|c|c|c|c|}\hline & x & & y & \\ \hline\end{array}$ and "x after y" – $\begin{array}{|c|c|c|c|c|}\hline & y & & x & \\ \hline\end{array}$), and there is a problem with the document. As some inconsistencies are only revealed through transitivities, superposing more than two strings will often present further issues.

Since the superposition operation would be used very frequently in any system that employed these strings for visualisation or computation, it is critical that it be made as efficient as possible.

3 Constraints for superposition of strings

Superposition in its simplest form is defined as the componentwise union of two strings of equal length:

$$\alpha_1\alpha_2\cdots\alpha_n \mathbin{\&} \alpha'_1\alpha'_2\cdots\alpha'_n \ := \ (\alpha_1\cup\alpha'_1)(\alpha_2\cup\alpha'_2)\cdots(\alpha_n\cup\alpha'_n) \tag{1}$$

for example

$$\begin{array}{|c|c|c|}\hline a & b & c \\ \hline\end{array} \mathbin{\&} \begin{array}{|c|c|c|}\hline a & a & d \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline a & a,b & c,d \\ \hline\end{array} \tag{2}$$

This is extended to languages (sets of strings) L and L':

$$L \mathbin{\&} L' \ := \ \bigcup_{n\geq 0}\{s \mathbin{\&} s' \mid s\in L_n,\ s'\in L'_n\} \tag{3}$$

where L_n and L'_n are the sets of strings of length n in L and L', respectively.

In order to extend this operation further to strings of unequal lengths, it should first be said that s and s' are *bc-equivalent* if they block compress to the same string, that is $\mathrm{bc}(s)=\mathrm{bc}(s')$. The inverse of the block compression operation may be used to introduce stutter in the strings which are to be superposed, and generate an infinite language of bc-equivalent strings:

$$\mathrm{bc}^{-1}(s)=\alpha_1^+\alpha_2^+\cdots\alpha_n^+ \qquad \text{if } s=\mathrm{bc}(\alpha_1\alpha_2\cdots\alpha_n) \tag{4}$$

These languages are then superposed, and the results are block compressed to give a finite set, the *asynchronous superposition* of s and s' (noting that $\mathrm{bc}(s)=\mathrm{bc}(s') \iff s'\in\mathrm{bc}^{-1}\mathrm{bc}(s)$):

$$s \mathbin{\&_*} s' \ := \ \{\mathrm{bc}(s'') \mid s''\in \mathrm{bc}^{-1}\mathrm{bc}(s) \mathbin{\&} \mathrm{bc}^{-1}\mathrm{bc}(s')\} \tag{5}$$

For example:

$$\begin{array}{|c|c|}\hline x & z \\ \hline\end{array} \mathbin{\&_*} \begin{array}{|c|c|c|}\hline x & y & z \\ \hline\end{array} = \{\begin{array}{|c|c|c|}\hline x & x,y & z \\ \hline\end{array}, \begin{array}{|c|c|c|c|}\hline x & x,y & x,z & z \\ \hline\end{array}, \begin{array}{|c|c|c|}\hline x & y,z & z \\ \hline\end{array}, \begin{array}{|c|c|c|c|}\hline x & x,y & y,z & z \\ \hline\end{array}, \begin{array}{|c|c|c|c|}\hline x & x,z & y,z & z \\ \hline\end{array}\} \tag{6}$$

Interestingly, the thirteen Allen interval relations given in

$$\mathcal{AR} \ := \ \{<,>,\text{d, di, f, fi, m, mi, o, oi, s, si},=\} \tag{7}$$

are represented by the asynchronous superposition

$$\begin{array}{|c|c|c|}\hline & a & \\ \hline\end{array} \mathbin{\&_*} \begin{array}{|c|c|c|}\hline & a' & \\ \hline\end{array} = \{\mathcal{S}_R(a,a') \mid R\in\mathcal{AR}\} \tag{8}$$

with each string $\mathcal{S}_R(a,a')$ featuring one relation $a\ R\ a'$, as shown in Table 1.

R	$a\,R\,a'$	$\mathcal{S}_R(a,a')$	R^{-1}	$a\,R^{-1}\,a'$	$\mathcal{S}_{R^{-1}}(a,a')$
$<$	a before a'	$\boxed{\ }\boxed{a}\boxed{\ }\boxed{a'}\boxed{\ }$	$>$	a after a'	$\boxed{\ }\boxed{a'}\boxed{\ }\boxed{a}\boxed{\ }$
m	a meets a'	$\boxed{\ }\boxed{a}\boxed{a'}\boxed{\ }$	mi	a met by a'	$\boxed{\ }\boxed{a'}\boxed{a}\boxed{\ }$
o	a overlaps a'	$\boxed{\ }\boxed{a}\boxed{a,a'}\boxed{a'}\boxed{\ }$	oi	a overlapped by a'	$\boxed{\ }\boxed{a'}\boxed{a',a}\boxed{a}\boxed{\ }$
d	a during a'	$\boxed{\ }\boxed{a'}\boxed{a,a'}\boxed{a'}\boxed{\ }$	di	a contains a'	$\boxed{\ }\boxed{a}\boxed{a',a}\boxed{a}\boxed{\ }$
s	a starts a'	$\boxed{\ }\boxed{a,a'}\boxed{a'}\boxed{\ }$	si	a started by a'	$\boxed{\ }\boxed{a',a}\boxed{a}\boxed{\ }$
f	a finishes a'	$\boxed{\ }\boxed{a'}\boxed{a,a'}\boxed{\ }$	fi	a finished by a'	$\boxed{\ }\boxed{a}\boxed{a',a}\boxed{\ }$
=	a equals a'	$\boxed{\ }\boxed{a,a'}\boxed{\ }$			

Table 1: Allen interval relations in strings

It should be noted that neither the basic ($\&$) or asynchronous ($\&_*$) forms of superposition pay attention to the semantics of the strings on which they operate. For instance, in (2), there is no way to determine the actual relation between c and d, given the strings $\boxed{a}\boxed{b}\boxed{c}$ and $\boxed{a}\boxed{a}\boxed{d}$. Similarly, the strings $\boxed{x}\boxed{z}$ and $\boxed{x}\boxed{y}\boxed{z}$ suggest a contradiction in (6), since each contains a different relation between x and z, only one of which could be veridical at once.

An upper bound on the length of the strings to be generated when using inverse block compression is established as $n + n' - 1$ in Woods et al. (2017), where n and n' are the respective lengths of input strings s and s'. However, while this limit is an obvious and necessary improvement on generating the infinite language $\mathrm{bc}^{-1}\mathrm{bc}(s)\ \&\ \mathrm{bc}^{-1}\mathrm{bc}(s')$ in (5), further constraints are required in order for superposition to truly be effective.

Some additional notation is presented here which will prove helpful in describing the present issue and its solution. The following string will be used for demonstrative purposes, with lt = "last Tuesday", js = "John sleeps", fa = "a fire alarm sounds":

$$\boxed{\ }\boxed{lt}\boxed{lt,js}\boxed{lt,js,fa}\boxed{lt,js}\boxed{lt}\boxed{\ } = \text{“John slept through the fire alarm last Tuesday”} \tag{9}$$

The *vocabulary* of a string s will be said to be the union of each of its components:

$$voc(s) := \bigcup_{i=1}^{n} \alpha_i \tag{10}$$

This makes s an $\mathrm{MSO}_{voc(s)}$-model,[2] interpreting each $a \in voc(s)$ as the set of string positions where a occurs. The vocabulary of (9) is $\{lt, js, fa\}$.

For any set A, the *A-reduct* of a string s is defined as the componentwise intersection of s with A (Fernando, 2016):

$$\rho_A(\alpha_1\alpha_2\cdots\alpha_n)\ :=\ (\alpha_1 \cap A)(\alpha_2 \cap A)\cdots(\alpha_n \cap A) \tag{11}$$

resulting in a string $\rho_A(s)$ with vocabulary $voc(s) \cap A$. For example, setting $A = \{lt, fa\}$, the A-reduct of (9) is the string $\boxed{\ }\boxed{lt}\boxed{lt}\boxed{lt,fa}\boxed{lt}\boxed{lt}\boxed{\ }$.

A string s will also be said to *project to* another string s' if the $voc(s')$-reduct of s block compresses to s':

$$\mathrm{bc}(\rho_{voc(s')}(s)) = s' \tag{12}$$

For example, a string projects to itself precisely if it is stutterless. Additionally, a language L can be said to project to a string s' if every string $s \in L$ projects to s'. As an $\mathrm{MSO}_{voc(s)}$-model, a string s *satisfies* $a\,R\,a'$ if s projects to $\mathcal{S}_R(a,a')$

$$s \models a\,R\,a' \iff \mathrm{bc}(\rho_{\{a,a'\}}(s)) = \mathcal{S}_R(a,a') \tag{13}$$

[2]See Libkin (2004) for a discussion of Monadic Second-Order Logic.

and "a is a bounded interval in s" if s projects to $\boxed{\,}\boxed{a}\boxed{\,}$. The string in (9) can be said to satisfy, for instance, "lt contains fa", since it projects to the string $\boxed{\,}\boxed{lt}\boxed{fa, lt}\boxed{lt}\boxed{\,}$ ("The fire alarm sounded at some point last Tuesday"). It also satisfies "lt contains js", and "js contains fa", as well as the inverses of these three relations.

In general, every string in $s \;\&_* \; s'$ will project back to both s and s', provided $voc(s) \cap voc(s') = \varnothing$. However, if this condition of disjoint vocabulary does not hold, then $s \;\&_* \; s'$ need not preserve the projections. For example, the superposition of "x before y" – $\boxed{\,}\boxed{x}\boxed{\,}\boxed{y}\boxed{\,}$ and "y before z" – $\boxed{\,}\boxed{y}\boxed{\,}\boxed{z}\boxed{\,}$ should, presumably,[3] result in a language containing exactly one string, namely $\boxed{\,}\boxed{x}\boxed{\,}\boxed{y}\boxed{\,}\boxed{z}\boxed{\,}$ ("x happened before y did, which happened before z"). This string projects to both of the strings which made it, preserving that original information, and further projects to "x before z" – $\boxed{\,}\boxed{x}\boxed{\,}\boxed{z}\boxed{\,}$, demonstrating one of the transitivities of the Allen interval relations i.e. the possible relation(s) between a and a'', given $a \; R \; a'$ and $a' \; R' \; a''$.

Asynchronous superposition in its current form, in fact, will produce a language of 270 strings, five of which project to $\boxed{\,}\boxed{x}\boxed{\,}\boxed{y}\boxed{\,}$, five which project to $\boxed{\,}\boxed{y}\boxed{\,}\boxed{z}\boxed{\,}$, and just one ($\boxed{\,}\boxed{x}\boxed{\,}\boxed{y}\boxed{\,}\boxed{z}\boxed{\,}$) which projects to both. Of these 270 strings, 245 will project to $\boxed{\,}\boxed{y}\boxed{\,}\boxed{y}\boxed{\,}$, which is plainly invalid by the earlier assertion that only finite intervals should be expected (i.e. for any fluent $a \in voc(s)$, s should project to $\boxed{\,}\boxed{a}\boxed{\,}$). By requiring that the resulting language of a superposition can project back to each of its "parent" strings, it is ensured that the original information is not lost, and allows for the calculation of the transitivities which are essential to temporal reasoning in Allen (1983) and Freksa (1992).

In Woods et al. (2017, p. 130), the potential results of a superposition are generated and then tested, checking each one for validity (using an algorithm based on matching string positions). While this does produce the correct output, it involves extensive overgeneration, even in the most basic of cases. Here an approach is presented which interleaves testing with generation, ensuring that only valid results are generated at all.

Θ is fixed as an infinite set of fluents, and $Fin(\Theta)$ as the set of finite subsets of Θ, such that any string s is in $Fin(\Theta)^*$. Given $\Sigma, \Sigma' \in Fin(\Theta)$, a function

$$\&_{\Sigma,\Sigma'} : (Fin(\Theta)^* \times Fin(\Theta)^*) \to 2^{Fin(\Theta)^*} \tag{14}$$

is defined, mapping a pair $(s, s') \in Fin(\Theta)^* \times Fin(\Theta)^*$ of strings to a set $s \;\&_{\Sigma,\Sigma'}\; s' \subseteq Fin(\Theta)^*$ of strings as follows:

$$\epsilon \;\&_{\Sigma,\Sigma'}\; \epsilon \;:=\; \{\epsilon\} \tag{15}$$

where ϵ is the empty string (of length 0)

$$\epsilon \;\&_{\Sigma,\Sigma'}\; s \;:=\; \varnothing \;\text{ for } s \neq \epsilon \tag{16a}$$

$$s \;\&_{\Sigma,\Sigma'}\; \epsilon \;:=\; \varnothing \;\text{ for } s \neq \epsilon \tag{16b}$$

and for $\alpha, \alpha' \in Fin(\Theta)$

$$\alpha s \;\&_{\Sigma,\Sigma'}\; \alpha' s' := \begin{cases} \{(\alpha \cup \alpha')s'' \mid s'' \in L(\alpha, s, \alpha', s', \Sigma, \Sigma')\} & \text{if } \Sigma \cap \alpha' \subseteq \alpha \text{ and } \Sigma' \cap \alpha \subseteq \alpha' \quad \dagger \\ \varnothing & \text{otherwise} \end{cases} \tag{17}$$

where $L(\alpha, s, \alpha', s', \Sigma, \Sigma')$ is

$$(\alpha s \;\&_{\Sigma,\Sigma'}\; s') \;\cup\; (s \;\&_{\Sigma,\Sigma'}\; \alpha' s') \;\cup\; (s \;\&_{\Sigma,\Sigma'}\; s') \tag{18}$$

(from which it follows that any string in $s \;\&_{\Sigma,\Sigma'}\; s'$ has length less than length(s) + length(s')). If $\Sigma = \Sigma' = \varnothing$, then (†) holds vacuously, and $\&_{\Sigma,\Sigma'}$ is functionally identical to the existant asynchronous superposition operation $\&_*$. Otherwise, (†) can be used to prevent those invalid superpositions which do not project to both s and s'.

[3]The assumption here being that all occurrences of a fluent symbol, whether appearing in one string or several, refer to the same unique event or time period.

Proposition 1. *For all $\Sigma, \Sigma' \in Fin(\Theta)$ and $s, s' \in Fin(\Theta)^*$, $s \,\&_{\Sigma,\Sigma'}\, s'$ selects those strings from $s \,\&_{\varnothing,\varnothing}\, s'$ which project to both the Σ-reduct of s and the Σ'-reduct of s'*

$$s \,\&_{\Sigma,\Sigma'}\, s' \;=\; \{s'' \in s \,\&_{\varnothing,\varnothing}\, s' \mid \mathrm{bc}(\rho_{voc(s)\cap\Sigma}(s'')) = \mathrm{bc}(\rho_\Sigma(s)) \textit{ and } \mathrm{bc}(\rho_{voc(s')\cap\Sigma'}(s'')) = \mathrm{bc}(\rho_{\Sigma'}(s'))\} \tag{19}$$

Corollary 2. *For all $s, s' \in Fin(\Theta)^*$ that are stutterless, if $\Sigma = voc(s)$ and $\Sigma' = voc(s')$, then $s \,\&_{\Sigma,\Sigma'}\, s'$ selects those strings from $s \,\&_{\varnothing,\varnothing}\, s'$ which project to s and s'*

$$s \,\&_{\Sigma,\Sigma'}\, s' \;=\; \{s'' \in s \,\&_{\varnothing,\varnothing}\, s' \mid \mathrm{bc}(\rho_\Sigma(s'')) = s \textit{ and } \mathrm{bc}(\rho_{\Sigma'}(s'')) = s'\} \tag{20}$$

Corollary 2 suggests that to preserve information under projection during superposition, *vocabulary constrained* superposition should be used:

$$s \,\&_{vc}\, s' \;:=\; s \,\&_{voc(s),voc(s')}\, s' \tag{21}$$

Below, (22) shows a short worked example for $\boxed{\ }\boxed{x}\boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{\ }\boxed{y}\boxed{z}\boxed{\ }$.[4]

$$(\boxed{\ } \cup \boxed{\ })(\boxed{\ }\boxed{x}\boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{y}\boxed{z}\boxed{\ } \cup \boxed{x}\boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{\ }\boxed{y}\boxed{z}\boxed{\ } \cup \boxed{x}\boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{y}\boxed{z}\boxed{\ }) \tag{22a}$$

$$(\boxed{\ })(\varnothing \cup (\boxed{x} \cup \boxed{\ })(\boxed{x}\boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{y}\boxed{z}\boxed{\ } \cup \boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{\ }\boxed{y}\boxed{z}\boxed{\ } \cup \boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{y}\boxed{z}\boxed{\ }) \cup \varnothing) \tag{22b}$$

$$(\boxed{\ }\boxed{x})(\varnothing \cup \varnothing \cup (\boxed{y} \cup \boxed{y})(\boxed{y}\boxed{\ } \,\&_{vc}\, \boxed{z}\boxed{\ } \cup \boxed{\ } \,\&_{vc}\, \boxed{y}\boxed{z}\boxed{\ } \cup \boxed{\ } \,\&_{vc}\, \boxed{z}\boxed{\ })) \tag{22c}$$

$$(\boxed{\ }\boxed{x}\boxed{y})(\varnothing \cup \varnothing \cup (\boxed{\ } \cup \boxed{z})(\boxed{\ } \,\&_{vc}\, \boxed{\ } \cup \epsilon \,\&_{vc}\, \boxed{z}\boxed{\ } \cup \epsilon \,\&_{vc}\, \boxed{\ })) \tag{22d}$$

$$(\boxed{\ }\boxed{x}\boxed{y}\boxed{z})((\boxed{\ } \cup \boxed{\ })(\boxed{\ } \,\&_{vc}\, \epsilon \cup \epsilon \,\&_{vc}\, \boxed{\ } \cup \epsilon \,\&_{vc}\, \epsilon) \cup \varnothing \cup \varnothing) \tag{22e}$$

$$(\boxed{\ }\boxed{x}\boxed{y}\boxed{z}\boxed{\ })(\{\epsilon\}) = \boxed{\ }\boxed{x}\boxed{y}\boxed{z}\boxed{\ } \tag{22f}$$

Compare the steps above with the procedure for asynchronous superposition. First, the padded forms of $\boxed{\ }\boxed{x}\boxed{y}\boxed{\ }$ and $\boxed{\ }\boxed{y}\boxed{z}\boxed{\ }$ must be generated, each of which are superposed together (in this case, 20×20 strings):

$$\{\boxed{\ }\boxed{\ }\boxed{\ }\boxed{x}\boxed{y}\boxed{\ } \,\&\, \boxed{\ }\boxed{\ }\boxed{\ }\boxed{y}\boxed{z}\boxed{\ }, \ldots, \boxed{\ }\boxed{x}\boxed{y}\boxed{\ }\boxed{\ }\boxed{\ } \,\&\, \boxed{\ }\boxed{y}\boxed{z}\boxed{\ }\boxed{\ }\boxed{\ }\} \tag{23}$$

Next, each of the resulting 400 strings is block compressed, to form a set of 53 possible strings:

$$\{\boxed{\ }\boxed{x,y}\boxed{y,z}\boxed{\ }, \ldots, \boxed{\ }\boxed{x,y}\boxed{\ }\boxed{y,z}\boxed{\ }\} \tag{24}$$

Finally, each of these strings is tested to ensure that it projects to both $\boxed{\ }\boxed{x}\boxed{y}\boxed{\ }$ and $\boxed{\ }\boxed{y}\boxed{z}\boxed{\ }$, reducing the set to $\{\boxed{\ }\boxed{x}\boxed{y}\boxed{z}\boxed{\ }\}$.

Below in Table 2 are some speed-tests comparing asynchronous superposition's generate-then-test approach against vocabulary constrained superposition, each run in the same test environment[5] with the same inputs. The correct strings are found by each algorithm, so the notable element here is simply the difference in time (the mean time of 1001 runs is given in milliseconds). Column 5 shows the percentage decrease in time to produce the final result from $\&_*$ to $\&_{vc}$, with a mean decrease over these six examples of 72.27%.

[4]It is worth mentioning that, although $\&_{vc}$ is written in each of (22), Σ and Σ' are fixed as $\{x, y\}$ and $\{y, z\}$, respectively, at the first step of this procedure.

[5]Node.js v10.0.0 (64-bit) on Ubuntu 16.04 using an Intel i7-6700 CPU with 16GB of memory.

s	s'	Time to $s\ \&_*\ s'$	Time to $s\ \&_{vc}\ s'$	Decrease in time
$\boxed{}\boxed{x}\boxed{}\boxed{y}\boxed{}$	$\boxed{}\boxed{y}\boxed{}\boxed{z}\boxed{}$	0.3207ms	0.0659ms	79.45%
$\boxed{}\boxed{x}\boxed{}\boxed{y}\boxed{}$	$\boxed{}\boxed{y}\boxed{}\boxed{x}\boxed{}$	0.3207ms	0.0180ms	94.39%
$\boxed{}\boxed{x}\boxed{}\boxed{y}\boxed{}$	$\boxed{}\boxed{x}\boxed{}\boxed{z}\boxed{}$	0.6124ms	0.2238ms	63.46%
$\boxed{}\boxed{x}\boxed{y}\boxed{}$	$\boxed{}\boxed{x}\boxed{}\boxed{z}\boxed{}$	0.1528ms	0.1059ms	30.69%
$\boxed{}\boxed{x}\boxed{}\boxed{y}\boxed{}$	$\boxed{}\boxed{u}\boxed{}\boxed{v}\boxed{}$	22.4016ms	5.3616ms	76.07%
$\boxed{}\boxed{w}\boxed{w,x}\boxed{x}\boxed{}\boxed{y}$	$\boxed{}\boxed{w}\boxed{}\boxed{z}\boxed{}$	4.7403ms	0.4955ms	89.55%

Table 2: Comparison of $\&_*$ and $\&_{vc}$

Although these particular inputs are, for the most part, quite basic and arbitrarily chosen, they do illustrate how preventing the problematic strings as they are generated, as in vocabulary constrained superposition, is generally quicker than generating all of the syntactically possible strings then testing them for validity, as asynchronous superposition does. The mean percentage decrease for all pairs of strings $s \in \mathcal{S}_R(x,y)$ and $s' \in \mathcal{S}_R(y,z)$, with $R \in \mathcal{AR}$, is 34.11%. With $s \in \mathcal{S}_R(x,y)$ and $s' \in \mathcal{S}_R(u,v)$ the mean decrease is 64.51%, and with both strings $s, s' \in \mathcal{S}_R(x,y)$, it is 74.28%, which seems to suggest that there are large gains in efficiency to be made in the cases when $voc(s) \cap voc(s') = \varnothing$ and when $voc(s) = voc(s')$.

4 Beyond Allen's relations

While treating all temporal propositions as finite, bounded intervals makes it straightforward to represent Allen's interval relations as strings, the situation is more complicated when the data is incomplete (as it most often is). If a superposition results in a language of size > 1, it is impossible to narrow that set down to a single, correct string without further information. In this scenario, we are left with a number of possible strings, each representing a different timeline of the events, only one of which is veridical.

Taking an example from TimeBank,[6] two of the TLINKs are converted to their string representations, featuring three distinct event instances ($ei1$, $ei2$, and $ei12$):

```
<TLINK relType="IBEFORE" eventInstanceID="ei1" relatedToEventInstance="ei2"/>
```

$$\text{“}ei1 \text{ meets } ei2\text{” – } \boxed{}\boxed{ei1}\boxed{ei2}\boxed{} \qquad (25)$$

```
<TLINK relType="AFTER" eventInstanceID="ei12" relatedToEventInstance="ei1"/>
```

$$\text{“}ei12 \text{ after } ei1\text{” – } \boxed{}\boxed{ei1}\boxed{}\boxed{ei12}\boxed{} \qquad (26)$$

Superposing these two strings gives a language with five strings, each of which projects to one of a disjunction of the possible Allen relations between $ei2$ and $ei12$ (see Table 3).

s	$\text{bc}(\rho_{\{ei2,ei12\}}(s))$	$ei2\ R\ ei12$
$\boxed{}\boxed{ei1}\boxed{ei2}\boxed{ei2,ei12}\boxed{}$	$\boxed{}\boxed{ei2}\boxed{ei12,ei2}\boxed{}$	$ei2$ finished by $ei12$
$\boxed{}\boxed{ei1}\boxed{ei2}\boxed{ei2,ei12}\boxed{ei2}\boxed{}$	$\boxed{}\boxed{ei2}\boxed{ei12,ei2}\boxed{ei2}\boxed{}$	$ei2$ contains $ei12$
$\boxed{}\boxed{ei1}\boxed{ei2}\boxed{ei12}\boxed{}$	$\boxed{}\boxed{ei2}\boxed{ei12}\boxed{}$	$ei2$ meets $ei12$
$\boxed{}\boxed{ei1}\boxed{ei2}\boxed{}\boxed{ei12}\boxed{}$	$\boxed{}\boxed{ei2}\boxed{}\boxed{ei12}\boxed{}$	$ei2$ before $ei12$
$\boxed{}\boxed{ei1}\boxed{ei2}\boxed{ei2,ei12}\boxed{ei12}\boxed{}$	$\boxed{}\boxed{ei2}\boxed{ei2,ei12}\boxed{ei12}\boxed{}$	$ei2$ overlaps $ei12$

Table 3: Results of superposition $s \in \boxed{}\boxed{ei1}\boxed{ei2}\boxed{}\ \&_{vc}\ \boxed{}\boxed{ei1}\boxed{}\boxed{ei12}\boxed{}$

It is impossible to determine which of the five is the correct relation for $ei2$ and $ei12$ given just the present data.

Given enough information, the exact relation between every fluent should be specifiable, achieving complete document closure and allowing the creation of a single string whose vocabulary is the set of the

[6]Document APW19980807.0261.tml

document's temporal propositions. The relation between any two entities e and e' would be determinable by applying the block compressed $\{e, e'\}$-reduct to the string, and seeing which Allen relation it projected to. However, in typical discourse (and the documents of TimeBank), this often cannot be done: two events may be presented ambiguously in their temporal ordering, either due to unclear wording, or due to the fact that the events were not put in any kind of relation to each other in the text. For example

The girl stopped singing when the music on the radio ended. (27)

It is evident that the music and the girl's singing ended at the same time, but it's unclear as to which started first, or if they started at the same time: the information just isn't there. It might be surmised that there is a probable correspondence between the music and the singing, and they likely started together, but it is not possible to be certain. Perhaps she was singing to the song before as well, or perhaps she only sang the last verse of the song.

Freksa (1992) proposed the use of semi-interval relations based on conceptual neighbourhoods to allow for description and reasoning about this kind of uncertainty, as well as coarser-level reasoning. A potential method for extending the expressivity of the strings discussed in this paper so as to allow for these semi-intervals is described below.

First, an interval a will now be said to be bounded in a string s if that string projects to $\boxed{pre(a)}\boxed{a}\boxed{post(a)}$, where pre($a$) and post($a$) are negations of a conjoined with a formula specifying that a occurs immediately to the right (in the case of pre(a)) or to the left (in the case of post(a)). Allowing non-atomic formulas such as these inside the string components does pose a risk of trivialising the work done by superposition, and so further study is perhaps required here before making a decision on what exactly should be permitted. For now, in any case, these symbols are taken as being allowed.

The example in Table 3 is repeated here, with these new bordering symbols made visible (for the sake of conciseness, $ei2$ and $ei12$ are abbreviated to x and y, respectively):

$x\ R\ y$	$prepost(s)$
x finished by y	$\boxed{pre(x), pre(y)}\boxed{x, pre(y)}\boxed{x, y}\boxed{post(x), post(y)}$
x contains y	$\boxed{pre(x), pre(y)}\boxed{x, pre(y)}\boxed{x, y}\boxed{post(x), y}\boxed{post(x), post(y)}$
x meets y	$\boxed{pre(x), pre(y)}\boxed{x, pre(y)}\boxed{post(x), y}\boxed{post(x), post(y)}$
x before y	$\boxed{pre(x), pre(y)}\boxed{x, pre(y)}\boxed{post(x), pre(y)}\boxed{post(x), y}\boxed{post(x), post(y)}$
x overlaps y	$\boxed{pre(x), pre(y)}\boxed{x, pre(y)}\boxed{x, y}\boxed{post(x), y}\boxed{post(x), post(y)}$

Table 4: Applying pre and post to s

From Figure 7, p21 of Freksa (1992), it can be seen that the list of Allen relations here corresponds to the Freksa relation "older", which has the constraint that the beginning point of x should be before the beginning point of y. Table 5 shows what happens when a block compressed $\{pre(x), pre(y)\}$-reduct is performed on the strings of Table 4:

$x\ R\ y$	$\mathrm{bc}(\rho_{\{pre(x), pre(y)\}}(prepost(s)))$
x finished by y	$\boxed{pre(x), pre(y)}\boxed{pre(y)}$
x contains y	$\boxed{pre(x), pre(y)}\boxed{pre(y)}$
x meets y	$\boxed{pre(x), pre(y)}\boxed{pre(y)}$
x before y	$\boxed{pre(x), pre(y)}\boxed{pre(y)}$
x overlaps y	$\boxed{pre(x), pre(y)}\boxed{pre(y)}$

Table 5: Performing a block compressed $\{pre(x), pre(y)\}$-reduct

It is clear from this that each of the Allen relations which make up the "older" Freksa relation project to $\boxed{pre(x), pre(y)}\boxed{pre(y)}$. In fact, only these five Allen relations will project to that string, and as such we can use it to characterise that relation, similar to how the strings in Table 1 characterise the various

Allen relations. Below are the characteristic projections for some of the more “simple” Freksa relations (note that some relations have multiple projections which could be considered characteristic, but just one is shown for each):

$$s \models a \text{ ol “older” } b \iff s \text{ projects to } \boxed{pre(a), pre(b)}\boxed{pre(b)}\boxed{\,} \tag{28a}$$

$$s \models a \text{ yo “younger” } b \iff s \text{ projects to } \boxed{pre(a), pre(b)}\boxed{pre(a)}\boxed{\,} \tag{28b}$$

$$s \models a \text{ sb “survived by” } b \iff s \text{ projects to } \boxed{\,}\boxed{post(a)}\boxed{post(a), post(b)} \tag{28c}$$

$$s \models a \text{ sv “survives” } b \iff s \text{ projects to } \boxed{\,}\boxed{post(b)}\boxed{post(a), post(b)} \tag{28d}$$

$$s \models a \text{ hh “head to head with” } b \iff s \text{ projects to } \boxed{pre(a), pre(b)}\boxed{\,} \tag{28e}$$

$$s \models a \text{ tt “tail to tail with” } b \iff s \text{ projects to } \boxed{\,}\boxed{post(a), post(b)} \tag{28f}$$

$$s \models a \text{ bd “born before death of” } b \iff s \text{ projects to } \boxed{pre(a)}\boxed{\,}\boxed{post(b)} \tag{28g}$$

$$s \models a \text{ db “died after birth of” } b \iff s \text{ projects to } \boxed{pre(b)}\boxed{\,}\boxed{post(a)} \tag{28h}$$

These relations are dubbed simple here as they can all be projected to a single string with no further extensions to the framework, which would permit condensing a language of results from a superposition into a single string which is characteristic of that language. Conversely, the remaining Freksa relations all involve either a conjunction or disjunction of constraints, and thus require further mechanics in order to fit with this string representation.

For example, the relation “contemporary of” can be defined as

$$\begin{aligned} s \models a \text{ ct “contemporary of” } b \iff s \text{ projects to } & \boxed{pre(a)}\boxed{\,}\boxed{post(b)} \\ \text{and } s \text{ projects to } & \boxed{pre(b)}\boxed{\,}\boxed{post(a)} \end{aligned} \tag{29}$$

and “precedes” can be

$$s \models a \text{ pr “precedes” } b \iff s \text{ does \textbf{not} project to } \boxed{pre(b)}\boxed{\,}\boxed{post(a)} \tag{30}$$

where “precedes” is effectively the opposite of “died after birth of”. This is obviously not ideal – while (29) is indeed a reduction from the nine Allen relations which correspond to “contemporary of”, a conjunction of two strings cannot easily be used as an input to another superposition. (30) is a bit better in that there’s only one string involved, but again, it’s difficult to superpose a negated string.

An option here is to expand again the symbols which can be boxed: allowing any pair of fluents a and a' to be conjoined within a string component such that for any α_i in a string s

$$a \wedge a' \in \alpha_i \iff a \in \alpha_i \text{ and } a' \in \alpha_i \tag{31}$$

for example

$$\rho_{\{x \wedge y\}}(\boxed{\,}\boxed{x}\boxed{x, y}\boxed{y}\boxed{\,}) = \boxed{\,}\boxed{\,}\boxed{x, y}\boxed{\,}\boxed{\,} \tag{32}$$

This allows for further Freksa relations to be represented as single strings

$$s \models a \text{ ct “contemporary of” } b \iff s \text{ projects to } \boxed{\,}\boxed{a \wedge b}\boxed{\,} \tag{33a}$$

$$s \models a \text{ pr “precedes” } b \iff s \text{ projects to } \boxed{pre(b)}\boxed{post(a) \wedge b}\boxed{\,} \tag{33b}$$

$$s \models a \text{ sd “succeeds” } b \iff s \text{ projects to } \boxed{pre(a)}\boxed{a \wedge post(b)}\boxed{\,} \tag{33c}$$

While these methods do seem to have some potential, further work needs to be done in order to fully integrate the concept of semi-intervals into this framework. For example, consider

$$\boxed{\,}\boxed{a}\boxed{b}\boxed{\,}\boxed{d}\boxed{\,} \;\&_{vc}\; \boxed{\,}\boxed{a}\boxed{\,}\boxed{c}\boxed{\,} \tag{34}$$

which produces a language of 25 strings. The relation "b older than c" can be derived from this result without too much difficulty, but the relation between c and d is more problematic. There is not enough data to suggest any of Allen's or Freksa's relations with any kind of certainty, and this total lack of information cannot be represented as anything remotely concise using strings. The question then arises as to what to do with this result.

The number of times each Allen relation occurred in the 25 strings could potentially be examined, and a guess hazarded based on the most frequently occurring one, but this approach would require careful testing to determine whether there is real merit. For example, of the 25 strings in (34), five of them (20%) in the resulting language suggest the relation "c before d", three (12%) for each of "overlaps", "contains", "finished by", and "meets", with the remaining eight Allen relations featuring in just one (4%) string each. This means 68% of those strings project to one of the five relations corresponding to "c older than d" – but there is no way to say for sure what relation there truly is between c and d without additional data.

Though it should be proved by experiment, it may be more beneficial in scenarios like this to simply leave the strings separate, and to not superpose them. Seeing two separate strings is plausibly more useful to an annotator than 25 alternate possibilities. Generalising from this, one could imagine that, rather than aiming for a single string representing the entire document's temporal structure, a number of maximal substrings are given instead, each of which contain as much information as can be made certain – that is, all strings are superposed with each other only under the condition that the language generated by superposition contains exactly one string (or group of strings corresponding to a single Freksa relation). In this way, the document's structure would still be visualised, but in chunks rather than a single timeline.

As an example, taking one of the smaller documents from TimeBank,[7] translate all of the TLINKs to strings (35), then superpose until there are no more options for superposition which produce a single-string language. The resulting set of strings (36) are the maximal substrings for the document.

$$\begin{aligned}\{\boxed{\,}\boxed{ei80}\boxed{t10}\boxed{\,}, \boxed{\,}\boxed{ei73}\boxed{t9}\boxed{\,}, \boxed{\,}\boxed{ei74}\boxed{ei73}\boxed{\,},\\ \boxed{\,}\boxed{ei75}\boxed{ei74}\boxed{\,}, \boxed{\,}\boxed{ei73, ei76}\boxed{\,}, \boxed{\,}\boxed{ei81}\boxed{ei80, ei81}\boxed{\,}\} \qquad (35)\end{aligned}$$

$$\{\boxed{\,}\boxed{ei75}\boxed{ei74}\boxed{ei73, ei76}\boxed{t9}\boxed{\,}, \boxed{\,}\boxed{ei81}\boxed{ei80, ei81}\boxed{t10}\boxed{\,}\} \qquad (36)$$

5 Conclusion

An efficient refinement was given for the superposition operation using an approach which interleaves generation with testing of valid strings, and a method was described for increasing the expressiveness of temporal strings to allow for some level of partial information based on semi-intervals.

There is still further experimentation to be done on how exactly to best handle cases with incomplete data, especially in large documents. Derczynski (2016) notes that a careful balance is needed between being exact in distinguishing relations, and not letting the set of relations available become too large. The question of precisely what to do when the result of a superposition cannot be condensed to a single string needs to be answered, and whether to allow for branching timelines in cases of non-determinism. Tooling is also in development to demonstrate the uses of the strings, in particular as a visualisation aid which may complement existing tools for the annotation process.

Acknowledgements

This research is supported by Science Foundation Ireland (SFI) through the CNGL Programme (Grant 12/CE/I2267) in the ADAPT Centre (`https://www.adaptcentre.ie`) at Trinity College Dublin. The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

[7]Document wsj_0006.tml

References

James F Allen. 1983. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, 26(11):832–843.

Leon Derczynski. 2016. Representation and Learning of Temporal Relations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1937–1948.

Tim Fernando. 2015. The Semantics of Tense and Aspect: A Finite-State Perspective. In S Lappin and C Fox, editors, *The Handbook of Contemporary Semantic Theory*, number August, pages 203–236. John Wiley & Sons.

Tim Fernando. 2016. On Regular Languages Over Power Sets. *Journal of Language Modelling*, 4(1):29–56.

Christian Freksa. 1992. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 54:199–227.

Leonid Libkin. 2004. Monadic Second-Order Logic and Automata. In *Elements of Finite Model Theory*, pages 113–140. Springer Berlin Heidelberg, Berlin, Heidelberg.

James Pustejovsky, Inderjeet Mani, Luc Belanger, Linda van Guilder, Robert Knippen, Andrew See, Jon Schwarz, and Marc Verhagen. 2003. Tango final report. In *ARDA Summer Workshop on Graphical Annotation Toolkit for TimeML, MITRE Bedford and Brandeis University*.

James Pustejovsky, Marc Verhagen, Roser Sauri, Jessica Littman, Robert Gaizauskas, Graham Katz, Inderjeet Mani, Robert Knippen, and Andrea Setzer. 2006. TimeBank 1.2 LDC2006T08. Web Download: `https://catalog.ldc.upenn.edu/LDC2006T08`. Philadelphia: Linguistic Data Consortium.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An International Standard for Semantic Annotation. In *LREC*, volume 10, pages 394–397.

TimeML Working Group. 2005. TimeML 1.2.1. A Formal Specification Language for Events and Temporal Expressions. `http://www.timeml.org/publications/timeMLdocs/timeml_1.2.1.html#tlink`.

Marc Verhagen, Robert Knippen, Inderjeet Mani, and James Pustejovsky. 2006. Annotation of Temporal Relations with Tango. *Proceedings of the 5th Languange Resources and Evaluation Confernece (LREC 2006)*, (May 2014):2249–2252.

Marc Verhagen. 2005a. Drawing TimeML Relations with T-BOX. In Graham Katz, James Pustejovsky, and Frank Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Internationales Begegnungs und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Marc Verhagen. 2005b. Temporal closure in an annotation environment. *Language Resources and Evaluation*, 39(2-3):211–241.

David Woods, Tim Fernando, and Carl Vogel. 2017. Towards Efficient String Processing of Annotated Events. In *Proceedings of the 13th Joint ISO-ACL Workshop on Interoperable Semantic Annotation (ISA-13)*, pages 124–133.