Formal Unifying Standards for the Representation of Spatiotemporal Knowledge

A report on ARLADA Task 02TA4-SP1-RT1
"Formalisms for Spatio-Temporal Reasoning"

Advanced Decision Architectures Alliance

Pat Hayes
IHMC

February 2004

Contents

1. Introduction

Much has been written in defense of various views on how best to describe time and change. Several schools of thought have emerged, ranging from the use of temporal modalities, a basic ontological distinction between *continuants* and *occurrents* (more intuitively, between things and processes) a basic distinction between 3-d entities and temporal processes, the situation calculus, and a unified 4-d view of spatiotemporal histories.

These various views seem to be sharply inconsistent with one another; they seem to be based on highly divergent intuitive ways of thinking about time and change, and the formalisms and ontological frameworks derived from them are usually thought of as being incompatible. They have been used as the foundations of various formal ontologies and in some cases put forward as basic ontological design principles, in direct opposition to other frameworks; the resulting debates have been intense but inconclusive. (One can derive a sense of the amount of debate by scanning the email archive of the IEEE Standard Upper Ontology Project, an on-going attempt to create a 'unified' ontological framework. See http://suo.ieee.org/) Similar divisions of opinion exist in how best to represent geographical and spatial information: in those cases the debates have been less intense, but still not arrived at a consensus.

We show here that all of these various philosophical and ontological frameworks can be treated as alternative decisions about the appropriate *syntactic* location of a temporal or spatial parameter in a descriptive formalism. In practice, in fact, this location can be left undecided, and the parameter can be adjusted dynamically by a generalized unification (matching) algorithm, which I call the ***trickledown unifier.*** This technique allows ontologies based on a variety of temporal and spatial philosophies and doctrines to be used within a common framework, without extensive re-writing or translation of one formalism into another.

The present Report describes the trickledown unifier, shows how this process works in a simple example, and explains the general conditions on parameters that are required in order for the process to work. Initially the discussion is restricted to temporal discourse, and then later we show that this technique generalizes to spatial and spaiotemporal representations in a uniform way.

2. Example

Consider a simple assertion involving time, such as “Bill was taller than his father and his aunt last year.” How should we formalize the content of such an assertion?

Technique A

We think of time as essentially a modality. Then the natural way to formalize any temporal assertion is to use a modal hybrid logic. The above example then becomes an assertion made in the present tense, but understood to be *asserted at* a time, considered to be an index for the modality. The basic sentence form here is a timeless assertion paired with a reference time. Using the symbol 'AT' to indicate the association of a sentence to a time, this example might look like this:

(Taller(Bill, father(Bill)) & Taller(Bill, aunt(Bill))) AT t in LastYear

This can be roughly understood as meaning "*At any time in last year, it would have been true to say that the Bill was taller than his father and his aunt*."

Technique B

Widely used in AI planning applications, the situation calculus expresses time by using *fluents*, i.e., properties and relations which are understood to be time-dependent. This is indicated by the inclusion of the time as an explicit argument of the relation, usually by convention the last argument. In fact, it is usual to assume that fluents take time-points as arguments rather than time-intervals, so the example would then be written with a quantified variable indicating the time:

forall t in LastYear, Taller(Bill, father(Bill), t) & Taller(Bill, aunt(Bill), t)

Note that the composer of such a formalization needs to know that Taller is a fluent but that Bill, father, and aunt are not. In general, the situation calculus assumes that the 'things' in the situation being described are unchanging (i.e., they are *continuants*), but may have changing properties. This technique is less well suited to describing processes or things that happen (i.e., *occurrents*), which must be thought of as state-changes and represented rather artificially as functions from states (times) to other states.

Technique C

By making a logical assertion about a temporal *history,* which is understood to be a 4-d 'slice' of a larger hypervolume of 4-dimensional space. In this case we 'reify' "*Bill during last year*" as a distinct entity, written as a term h(Bill, LastYear) using a special 'history' function h, and then assert some property of it:

Taller(h(Bill, LastYear), h(father(Bill), LastYear))
& Taller(h(Bill, LastYear), h(aunt(Bill), LastYear))

We have the option of rendering this in the point-based style analogous to that used in B:

forall t in LastYear, Taller(h(Bill, t), h(father(Bill), t)) & Taller(h(Bill, t), h(aunt(Bill), t))

but in this case it is an option. Indeed, we can use both together and have axioms describing their relationships, if required. In fact, the second form can be used to express a rather stronger statement, for instance, if we interpret Taller to mean an average over a time-interval. In contrast to Technique B, Technique C works equally well for processes as for things. In fact, it does not require us to make any principled distinction between them. It is also capable of expressing 'trend' information, so for example one can have a class of 'growing' histories and express 'Joe was getting taller all last year' in the form of a property of that particular Bill-history:

GettingTaller(h(Bill, LastYear))

This provides a considerable gain in expressiveness and makes for more efficient formalizations, but is often thought to conflict sharply with an important intuitive distinction between *things* (which endure through time) and *processes* (which occur at a time, or during a time-interval). It certainly requires a different set of intuitions to be comfortable with thinking of a motion, for example, as a kind of slope; or of a static 3-d world as merely a 'surface' between two 4-dimensional world-histories.

There are other approaches to temporal relativity in formal descriptions, but most of them can be viewed as variations on one of these three Techniques, perhaps expressed in different notational forms. For example, temporal relational databases typically conceptualize temporal information as adding one or more temporal parameters to relational tables, which can be viewed as a database formatting for a B-style formalization.

Of the three alternative Techniques, the first is often claimed to be the intuitively most approachable and easiest to understand, probably because it is closest to the way that time is indicated in natural language narrative descriptions. Unfortunately it is also the least flexible and formally the weakest and least expressive. Technique C has the opposite qualities: formally it is extremely effective, but is often rejected as highly unintuitive. The popularity of the situation-calculus style may be due to its being a workable compromise between these extremes, but this style has many well-known limitations

Discussion

The conceptualizations which give rise to these three alternative Techniques are sharply at odds with one another, and in some cases are based on different and strongly held philosophical and ontological views of the nature of time and events. For example, in his classic work on ontology, Peter Simons' classic work on ontology (1987) argues the case for what we have called technique A and rejects technique C as 'paradoxical' a

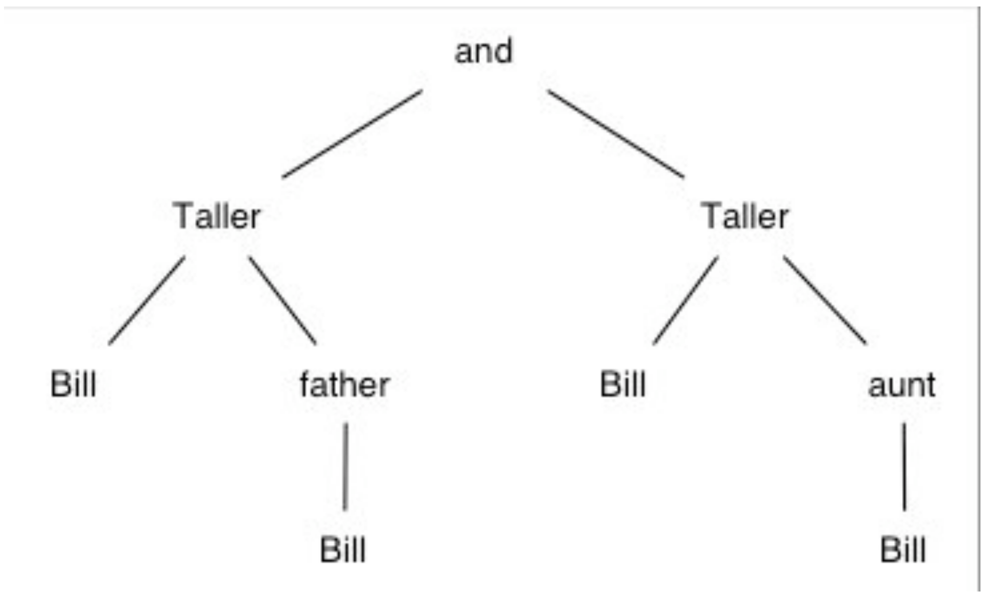
view that fits naturally with a philosophical tradition going back to Bolzano in the late 19th century, while other philosophers follow the ideas of A. N. Whitehead (1918), who argued that the universe was best conceptualized as consisting of systems of processes, which leads naturally to the technique C. The formal properties of the three Techniques also differ, as noted briefly above. As a result of this, each formalization style has its proponents who consider the choice of formalization to be based on matters of deeply-held principle or of long-established pragmatic convenience. The result has been a kind of Balkanization of techniques, several of which have been used as a basis for a formal standard. Thus, the Process Specification Language (PSL) standard [http://www.mel.nist.gov/psl/], which has had influence on several subsequent standardization efforts, is explicitly based on the situation-calculus state-function picture used in B, while the EPISTLE Core Model (ECM)
[http://www.fiatech.org/projects/idim/iso15926-2.htm] accepted as ISO 10303-221, is based on the 4-d model used in Technique C.

3. A Proposal for a Solution

Consideration of our simple example shows a striking similarity among the three Techniques, which becomes even more vivid if we parse the syntax of the formal sentences involved. Imagine a time-free assertion - in our case the plain assertion:

Taller(Bill, father(Bill)) and Taller(Bill, aunt(Bill))

and the obvious parsing of this in the form of a parse tree:



and consider the process of adding a temporal parameter as one of placing it at the root of the parse tree of this expression and allowing it to fall, or *trickle down*, to an appropriate level. Then the three techniques can be distinguished by what they consider to be this 'appropriate' syntactic level for temporal parameters to settle.

- Technique A keeps the parameter separate, allowing no insertion of the parameter into the expression.
- Technique B stops it at the level of a relational argument.
- Technique C allows it to go all the way down to individual names.

The particular notational devices used, such as AT and h(...) are then seen merely as means of 'attaching' a parameter to a syntactic location. If we use + as a common neutral syntactic device, the options appear respectively as:

(Taller(Bill, father(Bill)) & Taller(Bill, aunt(Bill))) +t

(Taller(Bill, father(Bill)) +t) & (Taller(Bill, aunt(Bill)) +t)

Taller(Bill+t, father(Bill)+t) & Taller(Bill+t, aunt(Bill)+t)

Note that the actual parameters are not affected by this choice of location; except that if a parameter is an interval, and the location where it finishes up requires a time-point, then we understand the result to be a quantified assertion over all points in the interval. (We will generalize this later.) All the logical apparatus of quantifying over intervals can be kept external to the expression whose temporal relativity is in question, however.

The proposal is basically to allow users to attach temporal parameters to expressions *in any combination*, basing the choice of attachment point on their favorite philosophical or pragmatic justifications, and achieving interoperability by allowing the inference machinery to match expressions by moving the parameters automatically. This is basically a very simple idea, and the bulk of the remaining development is concerned with fleshing out the details of this process and identifying the conditions under which it can reasonabbly be done. As we will see, a simple version is easy to describe and implement, but is applicable only under rather restricted assumptions, while a more general and theoretically powerful version of the system can be identified, which can be used under a wide variety of assumptions, but is likely to produce harder and less tractable inferential problems.

Since some attachment patterns are capable in principle of expressing temporal relationships which cannot be expressed in other disciplines - basically, the lower down one attaches the parameters, the more scope there is for expressing temporal subtleties - full intertranslation is not always possible in all directions; but the technique can extract as much common information as is logically possible, allowing maximal semantic interaction between superficially incompatible temporal formalizations.

4. Parameters and Temporal Sortings

As noted above, not all attachment points make sense. For example, Platonic entities such as numbers do not change their identities or properties with time, and are standardly omitted from any temporal parameterization. But also, for particular purposes it is often convenient to treat some entities as labile and others as fixed. In general, then, we will require the formal language being used to be *temporally sorted* by an assignment of the category 'temporal' to a subset of the names in the formal vocabulary. Such an assignment will be called a temporal sorting, or simply a *sorting*. Formally, it is simply a subset of the vocabulary, so that two or more sortings may be combined into a *unified sorting*. A sorting may assign temporality to an expression and to a subexpression of that expression. In this case we will say that the assignment to the subexpression is *below* the assignment to the expression itself, and the latter is *above* the former.

Here, 'temporal' means that a temporal parameter may be attached to any sub-expression with that symbol as its main symbol. Intuitively, it means that any expression which has this name as its main (top) symbol is understood to be referring in a time-sensitive way. If it is a relation symbol, then the truth of the relation may change with time; if it is a name of a thing, then whatever is being said about that thing may be true only at one time, or apply only to a temporal section of that thing's history.

- Technique A corresponds to the empty sorting which has no names classified as temporal. Technique B has only relational fluent names counting as temporal,
- Technique C classifies as temporal any name of a thing whose properties might vary, as well as any name of a 'temporally transparent' function such as father which satisfies father(x) + t = father(x + t).

As we have seen, two different temporal sortings may exist for the same vocabulary. The trickledown unifier allows expressions in a vocabulary to be unified even when two (or more) temporal sortings are in use.

5. Formal language

Suppose L is a logical vocabulary, which we will assume to be interpreted in a 'static' manner as referring to entities and relationships. In order to express temporal information we need a temporal vocabulary.

We will assume a 'basic' temporal theory T with terms indicating intervals and points, where points are understood as places where intervals start, finish and meet.

The time-order is then imposed by the intervals: one point is later than another when there is an interval with the first point as its start and the second point as its end. (This way of describing time has become a de facto standard in formal ontology work: it is used for example by PSL and by the OWL-S standard time ontology, among others.)

The details of the temporal theory need not concern us, other than it must somehow provide for a function on times which gives the *common time* for two times, satisfying the axioms

x < ct(s,t) iff (x<s and x<t)
not x < #

Here < is the part-of relationship (i.e., subinterval on intervals, inclusion between points and intervals, and identity between points) which is assumed to be reflexive and transitive on all time units, and # is the 'impossible' time used to indicate that two intervals do not overlap. We will also use $ to indicate the 'whole' time, i.e., the largest possible interval containing all other intervals:

x < $

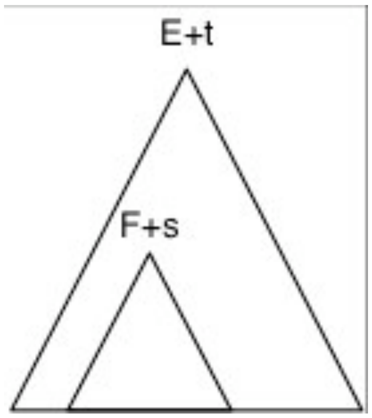
The unifier uses this function to determine the extent of temporal overlapping between times mentioned in a sentence, and the behavior of the algorithm is influenced by how easy it is to compute. In realistic cases, ct can often be computed efficiently by a process of consistency-checking using transitivity tables on interval relationships, or by boundary constraint analysis on endpoint ordering information.

Given an expression E in the language L and a term t in T, an expression of the form E+t is an *indexed expression*, whose intuitive meaning is that E is true during the time indicated by t, which is called the index of the expression. Note that E here may be a simple name, a denoting term such as father(Bill), a relational assertion or an entire sentence.

More generally, an indexed expression may have indexes attached to any of its sub-expressions, except that we will not permit indexing of the logical vocabulary (and, or, not, forall, exists) since its meaning is understood to be fixed. The effect would be the same if we allowed such indexing but required all such symbols to be indexed with a whole-time symbol indicating that they hold throughout all intervals.

Formally this requires that we invent a hybrid language L+T which uses the same syntactic construction rules as L but allows arbitrary expressions to be indexed by terms from T. I will omit the formal definitions here, but the construction is straightforward.

Suppose we have an expression E with a sub-expression F, each with an index attached:



for example

Running(father(Joe+s))+t .

We will say that such an expression *imposes* the time t *on* the time s. Intuitively, this asserts that something is true at, or during, one time but applies to (i.e., is true *of*) a thing which endures for another time; or, if E is a term, that a function applies at one time to an entity existing at another time. There are several ways to understand this, but for the present we will assume that this is meaningful only when t and s denote the *same* period of time. This assumption is referred to as *direct imposition*. This corresponds to a reading in which the 'inner' time indexing amounts to a selection of an appropriate sub-history of a longer history, which is a convention often adopted in Technique C. Note that direct imposition amounts to a point-like assumption about the expressions, where every expression denotes or is true of its arguments in a point-by-point fashion as time progresses, corresponding to the 'present-tense narrative' perspective adopted in case A. This could in fact be called the ‘narrative convention’ since it corresponds to the common device in natural language where a time-sequence is decribed by a ‘running present tense’ which moves with the events described.

Notice there is an assumption here that functions from entities to other entities preserve temporal indexing: in this example, father(Joe+s)=father(Joe)+s; that is, Joe's father during a period of Joe's life is the same as the father of Joe during that same period. This assumption therefore rules out some assertions which would be easy to write as single sentences using technique C, such as the “The motor of pump 17 was removed and sold a week later”, where the identification of the entity uses a time-period disjoint with the property being ascribed to it (of being sold when it was no longer the motor of the pump.)

Later we will discuss other ways of interpreting temporal imposition, which allow more complex and subtle temporal relationships to be handled involving relationships between things at different times. The basic idea of t-unification can be described using direct imposition, however, and many applications can be handled within this simple case very naturally.

Direct imposition means that any indexes of a sub-expression may be identified with the indexes of any super-expression: so all indexes can be 'moved' to the topmost level without loss of meaning. Thus all three of the Techniques can, under the assumption of direct imposition, be reduced to the simple, intuitive case of Technique A. The next section describes the simplest version of trickledown unification which is applicable under the assumption of direct imposition.

6. Unification With Direct Imposition

Many mechanical inference systems rely on a process of *unification* between expressions, which can be characterized as follows.

> Imagine laying the parse trees of the expressions on top of one another, starting at the root. If this assigns distinct non-variable labels to any node then the unification fails. If one tip node is labeled with a variable, then consider the sub-expression rooted at the corresponding node in the other expression. If it does not contain the variable, then *bind* that variable to the sub-expression and *substitute* that sub-expression for all other occurrences of that variable. If it does, then fail. Repeat until the expressions are identical or the unification fails.

The output of a successful unification is then the list of bindings to variables, and the *most general unifier*, or mgu, is the common instance of both expressions under the bindings. For example, the expressions

Older(father(x), x))   Older(y, John)

unify with output x/John and y/father(John) and most general unifier

Older(father(John), John))

Consider unifying two indexed expressions by following this algorithm on the unindexed expressions but also requiring that any two indexes in either expression must be unifiable if either is imposed on the other by the unification. We will call this *direct t-unification*: The mgu is the mgu of the unindexed expressions indexed with the mgu of the indices. For example, the expressions

Taller(x+s,father(x)+lastYear)   Taller(Bill,y)+t t-unify under the substitution x/Bill , t/lastYear , s/lastYear and y/father(Bill) with mgu

Taller(Bill,father(Bill))+lastYear

Note that the unification requires both s and t to be bound to one another and t to be bound to lastYear, thereby unifying all the indices to one, in virtue of the direct imposition requirement. Intuitively this act of unification can be understood as taking the assertions:

Someone at some time was taller than his father was last year at some time,
and
Bill was taller than someone

and merging them into a single conclusion.

It is clear that the t-unification process is of the same complexity as simple unification, which is known to be linear in the length of the expressions. Thus, t-unification provides an efficient technique for combining temporally sensitive information expressed in any of the three styles A, B or C; or indeed in a mixture of these styles: and the information can be expressed uniformly in the 'intuitive' A style.

T-unification by itself is not sufficient to support all of temporal reasoning. It handles the attachment of temporal parameters to other expressions, but we also need a temporal theory of the times themselves. We consider this in the next section.

7. Using Temporal Indices as Terms

Temporal constraints can be expressed by making unindexed assertions about the indices themselves. For example, the assertion,

Bill was taller than his father last year can be expressed simply by using a constant index, as above. But consider the more complex statement:

Bill became taller than his father some time last year.

This could be expressed thus:

exists t. t<lastYear & Taller(Bill,father(Bill))+t

where the body of the expression is a conjunction of a an indexed assertion and another assertion about the indices themselves, which we will call a *temporal constraints*. Notice that the temporal constraint is not indexed: under the direct imposition requirement, doing so would force t to be equal to lastYear. Thus a typical assertion will be a propositional composition (in this case, a conjunction) of temporal *constraints*, which refer to times but are not temporally indexed, and temporal *assertions*, which use temporal expressions as indices.

Efficient temporal planners and reasoners almost invariably use special reasoning techniques to handle purely temporal information, often organized as procedural call-outs 'attached' to the temporal relations. This separation of purely temporal constraints is useful in such a regime as it provides a clear distinction between normal inference and temporal constraint checking. One can then think of the unification process as a transmission of temporal information from assertions into constraints. We will make use of this intuition in the following sections to generalize the t-unification process to a more general imposition regime.

8. Indirect Imposition and Temporal Convexity

Direct imposition of indices is a very strict requirement, in that it assumes that all temporal indexing is exact. This is appropriate for many point- or instant-based temporal frameworks, or for applications such as calendar maintenance or scheduling which rely on precise numerical information; but for many purposes a more flexible regime is appropriate. In many formalisms, it is considered acceptable to assert a property of an enduring entity for some subinterval of the entity's lifetime, while still considering the longer lifetime to be the appropriate indexing interval for the entity itself, thereby using several distinct indexing conventions in one expression. For example, we might have the concept of *the person who was president during a given year*, and assert that this person spent a period of three days at Camp David, by writing an expression of the form:

```
exists t. t<yearN & long(t)=days(3) & where(president+yearN)+t=CampDavid
```

where the intention is not to assert that t is necessarily the entire year, only some subinterval of it. This *can* be expressed in a form that satisfies direct imposition, by separating the sub-expressions into pieces each with a consistent internal indexing:

```
exists t h. t<yearN & long(t)=days(3) & h=president+yearN & where(h)+t=CampDavid
```

but one needs to write rather more and, perhaps more significantly for realistic formalizations, one needs to *think* more carefully about what exactly is being said. Also notice that this is only locally direct, in that it is not possible to write it using a single indexed assertion, even though it seems to be intuitively about a single time period indicated by the variable t when the President was at Camp David.

We can generalize the technique to cover cases like this by making a common assumption about truth and times: That any temporal reference made relative to an interval will hold of all subintervals of that interval. That is, assuming that 'being true in I' means being true *throughout* I. This is often called the convexity assumption, and it can be formally expressed using our conventions:

if $J < I$ then $E+I$ implies $E+J$

We used this assumption implicitly when we translated an interval parameter into a simple universal quantifier over all points in the interval.

Not all temporal conventions in natural language satisfy convexity. For example, one might say *"Last week we drove all the way west from Boston to San Jose,"* without meaning to imply that the drive was never interrupted for sleep or refreshment. And it might even be true that for some periods of time while driving west, one is in fact driving east (because of an S-bend in the highway). As this example illustrates, however, one can usually find a sense of the words under which one can give a convex interpretation of any temporal subdivisions. For example, the phrase "driving west from Boston to San Jose" can be understood either in the sense of '*actually moving in a vehicle*' (not convex, except perhaps in a coast-to-coast rally) or in the sense of '*being engaged in a process of traveling using a vehicle*,' which could be understood to be true even while sleeping in a motel in Albuquerque, and hence could be convex. Similarly, "moving west" can be understood as '*moving while pointing in a westerly direction*' or as '*moving with a goal of arriving at a westward point*.' The latter is convex, the former likely not on roads through the Rockies.

Examples of properties that do not readily adapt to a convex reading include assertions of temporal averaging, for example '*the average speed is 60 mph*.' In a spatial analogue, this would include choropleth mapping conventions, for example, and assertions that imply a totality or completion, such as '*the chicken crossed the road*.'

If we assume convexity, then a 'mixed' indexing scheme does not introduce any ambiguity, so can be tolerated without complicating the language. The t-unification algorithm however needs to be modified. First, it is essential that variables are bound to *indexed* expressions, since the context of an assertion no longer completely determiines the intended temporal context of a binding; and second, when the unification process finds two indices s and t with s imposed on t, it *generates the additional temporal constraint* s<t rather than attempting to simply unify the indices. For example, given

```
where(x)+t=CampDavid
```

and

```
where(president+yearN)=y
```

to unify, this algorithm will unify with y/CampDavid, x/president+yearN and with the extra temporal constraint

```
t<yearN
```

The first change is simply a notational variation; the second is in line with the intuition described earlier of thinking of the unification process as one of communicating temporal constraints to a contraint checker. In fact in this case there is clearly no other option possible, since checking subinterval relationships can be complete only when done in the context of other information about temporal relationships that may be available. But it is likely to be less efficient. As is usual in formal inference, an increase in expressiveness and convenience comes with a cost in inferential complexity. However, it is easy to see that the behavior of the unification in this case simply automatically generates the extra constraints that would be part of the axiomatic description if direct imposition were used: so, one can think of this generalization of t-unification as a kind of automatic aid to axiomatization under a global assumption of temporal convexity.

Note also that this unifier is capable of some quite subtle implicit reasoning. For example, consider two expressions:

```
(where(x)=CampDavid)+yearN
```

forall s. duration(s)=year(1) implies exists t. t<s where(president+s)=CampDavid+t

The first expression might be phrased in English as the question *was anyone at Camp David during the yearN*? The second expression might be phrased as the assertion that any President spends some time at Camp David during each year of his tenure. The unifier can succeed by generating the binding x/president+s and the constraint t<yearN which, given the transitivity of <, in effect answers the question with "*the person who was the President during that year*." This may seem obvious, and indeed it is, but few mechanical logical reasoners could produce this answer without performing a search involving several reasoning steps.

Convexity is still a relatively restricted assumption about temporal expressions, and does not allow the formalization to take full advantage of the temporal flexibility of technique C. In the next section we consider a generalization of t-unification to a fully general temporal technique. However, there is no free lunch—this requires composers of formal representations to do quite a lot more work, and places a heavier burden on the constraint checker.

9. Oriented Indexing

One obtains convexity, as noted, by the assumption that a higher index must refer to a subinterval of any lower index, and that *every* name that can be used to label a non-leaf node in an expression (in the case of conventional logics, every relation and function name) is understood to 'handle' temporal indexicality in the same way. That is, any imposition from above indicates a time *within* the lifetime of any argument of the function. Similarly, the direct imposition regime corresponds to the assumption that all relations and functions apply exactly to their arguments in a 1:1, point-by-point basis. But other possibilities exist. If we have a relationship remembers, for example, so that remembers(x,y) means that x is remembering y, then the appropriate constraints for the indices in

remembers(x+t,y+s)+u

might be that u<t and s<t but s is entirely before u, on the grounds that one can only do the remembering while one exists, and can only remember things that happened while one was alive, but that one can only remember the past. It is possible to come up with examples of almost any pattern of constraints between argument and expression indexing.

In general, therefore, suppose that every function and relation name is provided with a set of temporal constraints which are required to hold between arguments or between arguments and the value of the expression, which we will call a *temporal orientation* for the symbol. We can represent a temporal orientation as an expression built using the temporal vocabulary T and variable names to stand for the arguments and expression. For example, the orienting constraint for remembers can be written as:

z=remembers(x,y): z<x & y<x & before(y,z)

Now, the t-unification algorithm applies as in the convex case, but when faced with the need to unify an index t imposed on another index s, the algorithm traces the symbols on the expression path from the attachment point of t to that of s, composes the orienting constraints for them and applies the resulting constraints to the indices. This effectively collects all the constraints that are implicit in the syntax of the expressions being unified which are required to be true in order for the unification to succeed.

The simpler cases mentioned prevously are all instances of this general technique. Direct imposition is the case where all symbols are oriented with a conjunction of identities, and convexity to the case where all symbols are oriented to be subintervals of their arguments.

Oriented indexing is capable of automatically generating quite elaborate temporal relationships and constraints, but the success of a reasoner is determined largely by its ability to properly process the resulting constraint language. Moreover, it can be hard to keep track of complex orienting constraints on a large vocabulary. For many applications, the simple direct imposition regime is adequate. However, for applications which require the full expressive power of technique C, a general oriented-constraint regime is capable of automatically and efficiently organizing the process of linking temporal planning constraints to a temporal description of a complex dynamic world.

The discussion so far has used examples and ideas from temporal representations, but the formal techniques require only that assertions be indexed with names on which a transitive 'sub' relation is defined. If we re-interpret the formalism so that indices refer to spatial locations and '<' refers to a sublocation relationship (i.e., to mean *inside* rather than *during)*, then the entire technique applies to *spatially* indexed assertions, which are then understood to be true *at a place* or to be referring to some spatially restricted area or location or part of an entity, identified by an index. We consider this in the subsequent sections.

10. Spatial, Temporal and Spatiotemporal Indexing

Examples of spatial indexing from natural language are less common than temporal examples, but examples include sentences which use spatial or geographical references to mean 'true at a place', as for example: "*In Louisiana, cases are decided using the Napoleonic code.*" The t-unification technique works in exactly the same way, both for direct and convex superposition, and adapts to spatial convexity in exactly the same way as it does to temporal convexity.

Spatial convexity corresponds to the global assumption that anything said about an areal location is thereby true of any sub-location, so that 'in' can be understood as meaning 'throughout'. The above example satisfies this constraint, for example, but not all spatial references do. Cartographic conventions are often discriminated on this basis. For example, as noted earlier, choropleth mapping conventions (in which a uniform rendering scheme is applied over an area such as a county or state to indicate some distributed property of that location) do not: the average income in a sublocation of a state may differ markedly from that of the state. Indeed, such conventions are often criticised on this basis, and conventions such as dasymetric mapping preferred on the grounds that they more accurately render spatial distributions, essentially by using a display mode which is visually convex.

Spatial indices are likely to require a more complex system of consistency checking for spatial assertions, but the basic advantages of allowing any style of spatial indexing of assertions still applies to the spatial case as to the temporal one. Both can be used together, in fact, provided that some notational device is used to distinguish spatial from temporal indices and to distinguish subinterval (during) from sublocation (inside).

Maintaining separate temporal and spatial indexing essentially treats the spatial and temporal indexings as being orthogonal. While a common assumption, there are applications for which it is more useful to combine them into a single *spatiotemporal* indexing scheme.

For example, a dynamic geographically distributed phenomenon such as a tropical storm or a complex military operation of troop deployment can be viewed as consisting of spatially extended components whose location and extent may be changing with time and which may bear dynamically changing relationships to one another. Constraints such as '*the storm surge will reach the coast within the next 5 hours*' are hard to express without mixing spatial and temporal information in non-orthogonal ways.

Fortunately, the t-unification technique adapts smoothly to this more general case, provided that the basic indices are understood to refer to coherent spatiotemporal 'regions' or histories, and the 'subinterval' relationship is generalized to a sub-history relationship which might be given an English gloss as '*within'*, where this is understood to refer to both temporal and spatial inclusion. This does however require the reasoner to be capable of checking consistency of spatiotemporal constraints.

11. Conclusions and Recommendations

Information processing systems represent information about spatial and temporal relations in a variety of ways. The present effort has been aimed at developing formalisms that would allow automatic "translation" among alternative frameworks ("ontologies") for denoting spatiotemporal relations.

This work has yielded a solution to the problem of unifying spatiotemporal representations. Readers should note that this does not solve all of the problems of spatiotemporal inference. What it does do is provide a framework within which spatiotemporal reasoning can be performed, unifying what had heretofore been regarded as incompatible approaches.

This is of potential significance for application in the Semantic Web and in distributed computing, but is also of potential significance for domains where different technologies, designed for different purposes, need nevertheless to be able to share information or data, process the information at a meaningful level, and ideally perform inference operations on the information.

It is recommended that a next step in this work be to collaborate with researchers at the ARL Computational and Information Sciences Directorate, to determine ways in which the general logical analyses presented here can be concretized, tailored, and applied to Army current and future needs.

12. References

Simons, P. (1987). *Parts: A study in ontology*. Oxford: Clarendon Press.

Whitehead, A. N. (1918). *Process and reality*. Oxford: Oxford University Press.