# TYPE THEORY AND SEMANTICS IN FLUX*

Robin Cooper

**DRAFT September 26, 2010**

## 1 INTRODUCTION

A frequent assumption in computational and corpus linguistics as well as theoretical linguistics is that words are associated with a fairly small set of meanings, statically defined in a lexical resource. [Jurafsky and Martin, 2009, Chap. 19 is a standard textbook reference presenting this kind of view.] This view is challenged by work in the psychology of language [Clark and Wilkes-Gibbs, 1986; Garrod and Anderson, 1987; Pickering and Garrod, 2004; Brennan and Clark, 1996; Healey, 1997, among others] where dialogue participants are regarded as creating meaning on the fly for the purposes of particular dialogues and this view has been taken up by recent approaches to dialogue semantics [Larsson, 2007b; Larsson, 2007a; Larsson and Cooper, 2009; Cooper and Larsson, 2009; Ginzburg, forthcoming]. [Cooper, 2010a] argues that a view of lexical meaning in flux is important for the lexicon in general, not just for the analysis of dialogue. Here we will explore the philosophical underpinning of this argument, in particular the kind of type theory with records (TTR) that we propose [Cooper, 2005a; Cooper, 2005b; Ginzburg, forthcoming].

The philosophical argument relates to two views of language that create a tension in the philosophy of language that has essentially remained unresolved since the middle of the last century. The conflict is represented in the contrast between early and late Wittgenstein, that is, the view represented in the *Tractatus* [Wittgenstein, 1922] as opposed to *Philosophical Investigations* [Wittgenstein, 1953]. We can think of the positivistic view of early Wittgenstein as somewhat related to the view of natural languages as formal languages expressed by Montague [Montague, 1974], even though Montague was reacting against the positivistic view of natural language as imprecise and informal. Montague's application of formal language techniques to natural language does, however, give the impression of

*I am grateful to Raquel Fernández, Jonathan Ginzburg, Ruth Kempson, Staffan Larsson and Bengt Nordström for comments. I am particularly grateful to Tim Fernando for very detailed comments that led to a major rewriting. This work was supported in part by a grant from Vetenskapsrådet, Library-based Grammar Engineering (2005-4211), The Swedish Bank Tercentenary Foundation Project P2007/0717, Semantic Coordination in Dialogue and VR project 2009-1569, Semantic analysis of interaction and coordination in dialogue (SAICD).

Handbook of the Philosophy of Science. Volume 14: Philosophy of Linguistics
Volume editors: Ruth Kempson, Nicholas Asher and Tim Fernando. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

natural languages as being regimented with meanings determined once and for all by an interpretation. This is a view which is very different from that of the late Wittgenstein who talked in terms of language games and the creation of public language for specific purposes. [Cooper and Ranta, 2008] represents a sketch of an attempt to take something like the late Wittgenstein view without throwing away the immense advances that were made in twentieth century semantics by the application of Montague's techniques. The idea there is that natural languages are to be seen as toolboxes (*resources*) that can be used to create limited languages for use in particular language games in the sense of late Wittgenstein. These limited special purpose languages may be formal in the sense that Montague had in mind. We will argue, however, that there is a lot of linguistic interest in trying to discover not only how natural languages provide these formal languages but also how agents using the language apply and develop these resources which are constantly in a state of flux as we use the language. We will argue that our particular kind of type theory is appropriate for such an analysis whereas the kind of semantics of the classical model theoretic approach represented by Montague does not provide us with enough structure to capture the notions of variation in meaning that appear to be necessary.

When people talk to each other they create new language suitable for discussing the subject matter they are addressing. Occasionally, people will create entirely new words to express a new concept that they are trying to convey to their interlocutor. More often, though, they will use a previously existing word but with a modified meaning to match the new concept. In order to analyze this we need an approach to meaning in terms of structured objects that can be modified. Sometimes innovation is asymmetric in the sense that the speaker uses a word in a way that is not innovative for her but the hearer either does not know the word at all or has not previously heard the word associated with the particular meaning intended by the speaker. The hearer processes and learns the new way of using the word by modifying the meaning he had previously associated with the word or, if the word is entirely new to him, possibly by modifying a similar meaning he associates with a different word. In order to analyze this we need an approach to meaning which allows a general notion of a similarity measure on meanings. This, like the modification of meaning associated with the learning of the innovative meaning, can be achieved by treating meanings in terms of structured objects where we can see, for example, how many components a pair of structured meanings share.

The classical notion of meaning from model theoretic semantics is that meaning is a function from possible worlds and contexts to denotations derived from the domain of the model. We will argue that record types provide us with feature structure like objects which easily admit similarity measures and structural modifications because they are structured into fields containing a label and a value. Similarity measures can be created by comparing fields in two objects and objects can be modified by adding or deleting fields or changing the value provided for a particular field.

The general view of language in which our discussion will be cast is that of

language as action, speech events that can cause changes in the mental states of dialogue participants during the course of linguistic interaction. This view of language, though it might be seen as contrasting with the kind of formal language view presented by Montague [Montague, 1974] or even the general Chomskyan tradition, is not new. Apart from Wittgenstein, it has roots, for example, in speech act theory [Austin, 1962; Searle, 1969]. An early attempt to take an action or event-based view of all aspects of compositional semantics is [Barwise and Perry, 1983]. Two recent works that develop a linguistic view of interaction are [Ginzburg, forthcoming] and [Linell, 2009], although these two books take very different approaches and have almost no overlap in the literature they refer to.

A frequent complaint against classical formal semantics is that it has nothing to say about the details of word meaning. If you have a superficial analysis of word meaning then it can appear that uses of words in many different situations have the same meaning. We shall argue that as you examine the details of word meaning, we see that the situation is much more like that proposed in the late Wittgenstein, that is, word meaning varies according to the use to which it is put in a particular communicative situation. In the following sections we will pursue the example discussed in [Cooper, 2010a] and show in detail how to construct a type theory to support the analysis we propose, based on a notion of frame deriving from Frame Semantics [Fillmore, 1982; Fillmore, 1985]. In what follows many sections are marked with a star. These sections may be omitted on first reading. By following the unstarred sections the reader will obtain an emended version of [Cooper, 2010a]. Readers who dip into the starred sections will in addition get a technical account of what is discussed in the unstarred sections as well as some more philosophical background. The unstarred sections occur at the beginning of the main sections. Section 2 is concerned with how we can use TTR to represent frames in the sense of Fillmore's frame semantics and developing the type theory we need to do this. Section 3 is concerned with how such frames could be exploited in the compositional semantics of verbs. Section 4 shows how this analysis can be used to solve a classical puzzle from formal semantics: the Partee puzzle concerning the rising of temperature and price. Section 5 looks more deeply into the lexical semantics of a single verb *rise* using Fernando's string theory of events. Here we discover that looking more deeply at the lexical meaning of this verb suggests that meaning varies from situation to situation and that there is always an option for creating new meaning. In section 6 we place this observation in the context of the view of coordination that has been developed by Larsson. Finally in section 7 we draw some conclusions.

## 2 FRAMES

### *2.1 Representing frames in TTR*

Frame semantics was introduced in Fillmore's classic paper [Fillmore, 1982]. We will use semantic objects which are related to the frames of FrameNet.[1] An important part of our proposal will be that these objects can serve as the arguments to predicates. We will use record types as defined in TTR ( [Cooper, 2005a; Cooper, 2005b; Ginzburg, forthcoming]) to characterize our frames. The advantage of records is that they are objects with a structure like attribute value matrices as used in linguistics. Labels (corresponding to attributes) in records allow us to access and keep track of parameters defined within semantic objects. This is in marked contrast to classical model theoretic semantics where semantic objects are either atoms or unstructured sets and functions.

Consider the frame `Ambient_temperature` defined in the Berkeley FrameNet[2] by "The Temperature in a certain environment, determined by Time and Place, is specified". Its core frame elements are given in (1).

(1) **Attribute** The temperature feature of the weather

**Degree** A modifier expressing the deviation of the Temperature from the norm

**Place** The Place where it is a certain Temperature

**Temperature** A quantity or other characterization of the Temperature of the environment

**Time** The Time during which an ambient environment has a particular Temperature

To make things of a manageable size we will not include all the frame elements in our representation of this frame. (We have also changed the names of the frame elements to suit our own purposes.) We will say that an ambient temperature frame is a record of type (2).

(2)
$$\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{e-time} & : & \mathit{Time} \\ \text{e-location} & : & \mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}} & : & \text{temp\_at\_in(e-time, e-location, x)} \end{bmatrix}$$

We will call this type *AmbTemp*. It is a set of four fields each consisting of a *label* (to the left of the colon) and a type (to the right of the colon). A record of type *AmbTemp* will meet the following two conditions:

[1] `http://framenet.icsi.berkeley.edu/`
[2] accessed 25th Oct, 2009

- it will contain *at least* fields with the same labels as the type (it may contain more)
- each field in the record with the same label as a field in the record type will contain an object of the type in the corresponding field of the record type. (Any additional fields with different labels to those in the record type may contain objects of any type.)

Types constructed with predicates such as 'temp_at_in' have a special status in that they can be *dependent*. In (2) the type in the field labelled '$c_{temp\_at\_in}$' depends on what you choose for the other three fields in the frame. Intuitively, we can think of such types formed with a predicate like 'temp_at_in' as types of objects which prove a proposition. What objects you take to belong to these types depends on what kind of theory of the world you have or what kind of application you want to use your type theory for. Candidates would be events, states or, in this case, thermometer or sensor readings.

The notions that we need to define in our type theory in order to achieve this are:

- *basic types*, such as *Ind*, *Time* and *Loc*
- *complex types* constructed with *predicates*
- *record types* based on basic types and complex types with predicates

We will see below that our construction of record types will in addition require us to introduce *function types* and a type *Type* of types which will lead us to *stratify* our type system. We will begin by presenting some philosophical background for the type theory.

## *2.2 Type theory, mathematics and cognition

The philosophical foundation of type theory (as presented, for example, by [Martin-Löf, 1984]) is normally seen as related to intuitionism and constructive mathematics. It is, at bottom, a proof-theoretic discipline rather than a model-theoretic one (despite the fact that model theories have been provided for some type theories). However, it seems that many of the ideas in type theory that are important for the analysis of natural language can be adopted into the classical set theoretic framework familiar to linguists from the classical canon of formal semantics starting from [Montague, 1974]. There is a risk in pushing this line of alienating both the type theorists (who feel that the philosophical essence of type theory is being abandoned) and the linguists (who tend to feel that if one is going to move in the type theory direction then one should probably be doing proof theory rather than model theory). Ultimately, the line between a proof theoretical approach and a model-theoretic approach that advocates structured semantic objects can be a hard one to draw when viewed from the perspective of a theory of natural language or human cognition. Both approaches are advocating the need for more structured

objects than are provided by classical model theory, objects whose components can be manipulated by formal processes which are meant to model agents' cognitive processes. In this section we will attempt to present a philosophical view of our type theory as an important component in a theory of cognition.

The notion of type that we are discussing is more general than the notion of type found, for example, in Russell's theory of types as it was adapted to Montague's semantics, that is, entities, sets, sets of sets, function from objects of one type to another, and so on. The kind of types we are discussing here correspond to what might be called properties in other theories. Types correspond to pretty much any useful way of classifying things.

While perception and typing are at the core of cognitive processing an important feature of cognitive systems is the ability to consider alternative typings which have not be observed. While we perceive $a$ to be of type $T_1$ it is perhaps nevertheless conceivable that $a$ could have been of type $T_2$. This leads us to construct modal type systems with alternative assignments of objects to types.

In addition to basic types, cognitive agents perceive the world in terms of states and events where objects have properties and stand in relations to each other – what [Barwise and Perry, 1983] called situations. Thus we introduce types which are constructed from predicates (like 'hug') and objects which are arguments to this predicate like $a$ and $b$. We will represent such a constructed type as hug($a$,$b$). What would an object belonging to such a type be? According to the type-theoretic approach introduced by Martin-Löf it should be an object which constitutes a proof that $a$ is hugging $b$. For Martin-Löf, who was considering mathematical predicates, such proof objects might be numbers with certain properties, ordered pairs and so on. [Ranta, 1994] points out that for non-mathematical predicates the objects could be events as conceived by [Davidson, 1980]. Thus hug($a$,$b$) can be considered to be an event or a situation type. In some versions of situation theory [Barwise, 1989; Seligman and Moss, 1997], objects (called *infons*) constructed from a relation and its arguments was considered to be one kind of situation type. Thus one view would be that these kinds of types are playing a similar role in type theory to the role that infons play in situation theory.

These types play a role in the "propositions as types" dictum which comes from type theory. If hug($a$,$b$) is the type of events where $a$ hugs $b$ then the sentence "$a$ hugs $b$" will be true just in case this type is non-empty, that is, just in case there is an event where $a$ hugs $b$. The type can function as the theoretical object corresponding to the informal notion of proposition. It is "true" just in case it is non-empty.

An important aspect of human cognition is that we seem to be able to treat the types themselves as if they were objects. This becomes apparent when we consider attitude predicates like 'believe'. In classical model theoretic semantics we think of *believe* as corresponding to a relation between individuals and propositions. In our type theory, however, we are subscribing to the "propositions as types" view. It then follows that the second argument to the predicate 'believe' should be a type. That is, we should be able to construct the type believe($c$, hug($a$,$b$))

corresponding to *c believes that a hugs b*. We thus create intensional type systems where types themselves can be treated as objects and belong to types. Care has to be taken in constructing such systems in order to avoid paradoxes. We use here a standard technique known as stratification [Turner, 2005]. We start with a basic type system and then add higher order levels of types. Each higher order includes the types of the order immediately below as objects. In each of these higher orders $n$ there will be a type of all types of the order $n-1$ but there is no ultimate "type of all types" – such a type would have to have itself as an object.

We will argue below that it is very important that the complex types we introduce are structured which enables them to be compared and modified. This is what makes it possible to account for how agents exploit and adapt the resources they have as they create new language during the course of interaction. It is not quite enough, however, simply to have objects with components. We also need a systematic way of accessing these components, a system of labelling which will provide us with handles for the various pieces. This is where the record types of TTR come in. There is a large literature on type theories with records in computer science, for example, [Tasistro, 1997; Betarte, 1998; Betarte and Tasistro, 1998; Coquand *et al.*, 2004]. Our notion of record type is closely related to those discussed in this literature, though (like the rest of TTR) couched in rather different terms. For us a record type is a set of fields where each field is an ordered pair of a label and a type (or a pair consisting of a dependent type and a sequence of path names corresponding to what the type is to depend on). A record belonging to such a type is a set of fields which includes fields with the same labels as those occurring in the type. Each field in the record with a label matching one in the type must contain an object belonging to the type of the corresponding field in the type.

It is an important aspect of human cognition that we not only appear to construct complex cognitive objects out of smaller ones as their components but that we also have ways of accessing the components and performing operations like substitutions, deletions and additions. Cognitive processing also appears to depend on similarity metrics which require us to compare components. Thus labelling or the provision of handles pointing to the components of complex objects is an important part of a formal theory of human cognition and in TTR it is the records and record types which do this work for us.

The importance of labelling has been reflected in the use of features in linguistic theorizing ranging from the early Prague school [Trubetzkoy, 1939] to modern feature based grammar [Sag *et al.*, 2003]. It appears in somewhat different form in the use of discourse referents in the treatment of discourse anaphora in formal semantics [Kamp and Reyle, 1993]. In [Cooper, 2005b] we argue that record types can be used both to model the feature structures of feature based grammar and the discourse representation structures of discourse representation theory. This is part of a general programme for developing TTR to be a general type theory which underlies all our linguistic cognitive processing. In fact, what we would like to see in the future is a single type theory which underlies all of human cognitive

processing.

In contrast to the statement of TTR in [Cooper, 2005a] we attempt here to present interacting modules which are not that complex in themselves. Nevertheless, knowing exactly what you have got when you put everything together is not an entirely trivial matter. It would be nice from a logical point of view if human cognition presented itself to us in neat separate boxes which we could study independently. But this is not the case. We are after all involved in the study of a biological system and there is no reason in principle why our cognitive anatomy should be any simpler than our physical anatomy with its multiplicity of objects such as organs, nerves, muscles and arteries and complex dependencies between them, though all built up on the basis of general principles of cell structure and DNA. Compared with what we know about physical anatomy, TTR seems quite modest in the number of different kinds of objects it proposes and the interrelationships between them.

## *2.3 Basic types

The simplest type system we will introduce has no complex types. All the types are atoms (that is they are objects which are not constructed from other objects in the system) and the of-type relation is determined by a function which assigns sets of objects to types. We will call this a system of basic types.

A *system of basic types* is a pair:

$$\mathbf{TYPE}_B = \langle \mathbf{Type}, A \rangle$$

where:

1. **Type** is a non-empty set
2. $A$ is a function whose domain is **Type**
3. for any $T \in \mathbf{Type}$, $A(T)$ is a set disjoint from **Type**
4. for any $T \in \mathbf{Type}$, $a :_{\mathbf{TYPE_B}} T$ iff $a \in A(T)$

Central to type theory is the notion of judgements that an object $a$ is of a type $T$ (in symbols $a : T$). We see this as being fundamentally related to perception. When we perceive objects in the world, we perceive them as belonging to a particular type (or perhaps several types). There is no perception without some kind of judgement with respect to types of the perceived object. When we say that we do not know what an object is, this normally means that we do not have a type for the object which is narrow enough for the purposes at hand. I trip over something in the dark, exclaiming "What's that?", but my painful physical interaction with it through my big toe tells me at least that it is a physical object, sufficiently hard and heavy to offer resistance to my toe. The act of perceiving an object is perceiving it *as* something. That "something" is a type.

The notion of type judgements yields a type theory with two domains: one domain for the objects and another domain for the types to which these objects belong. Thus we see types as theoretical entities in their own right, not, for example, as collections of objects. Diagrammatically we can represent this as in Figure 1 where object $a$ is of type $T_1$.

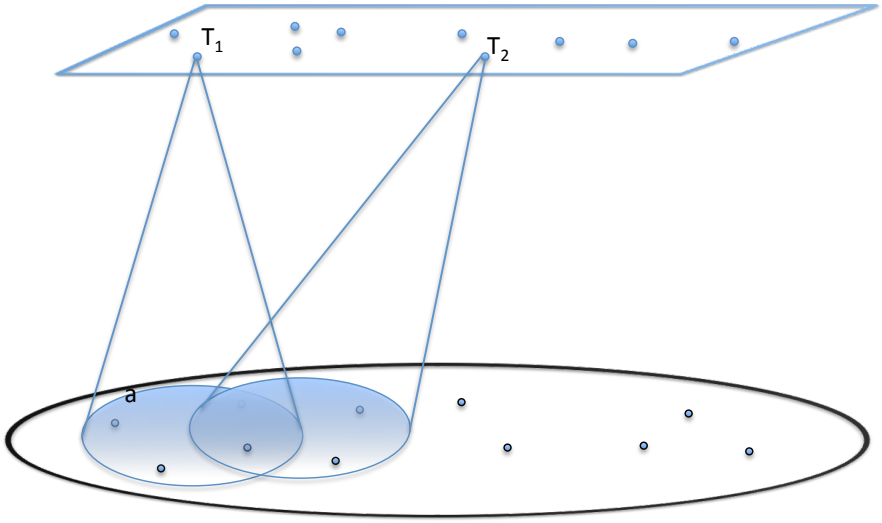


Figure 1. System of basic types

## *2.4 *Complex types*

We start by introducing the notion of a predicate signature.

A *predicate signature* is a triple

$$\langle \textbf{Pred}, \textbf{ArgIndices}, \textit{Arity} \rangle$$

where:

1. **Pred** is a set (of predicates)
2. **ArgIndices** is a set (of indices for predicate arguments, normally types)
3. *Arity* is a function with domain **Pred** and range included in the set of finite sequences of members of **ArgIndices**.

A *polymorphic predicate signature* is a triple

$$\langle \textbf{Pred}, \textbf{ArgIndices}, \mathit{Arity} \rangle$$

where:

1. **Pred** is a set (of predicates)
2. **ArgIndices** is a set (of indices for predicate arguments, normally types)
3. *Arity* is a function with domain **Pred** and range included in the powerset of the set of finite sequences of members of **ArgIndices**.

A *system of complex types* is a quadruple:

$$\textbf{TYPE}_C = \langle \textbf{Type}, \textbf{BType}, \langle \textbf{PType}, \textbf{Pred}, \textbf{ArgIndices}, \mathit{Arity} \rangle, \langle A, F \rangle \rangle$$

where:

1. $\langle \textbf{BType}, A \rangle$ is a system of basic types
2. $\textbf{BType} \subseteq \textbf{Type}$
3. for any $T \in \textbf{Type}$, if $a :_{\langle \textbf{BType}, A \rangle} T$ then $a :_{\textbf{TYPE}_C} T$
4. $\langle \textbf{Pred}, \textbf{ArgIndices}, \mathit{Arity} \rangle$ is a (polymorphic) predicate signature
5. If $P \in \textbf{Pred}$, $T_1 \in \textbf{Type}, \ldots, T_n \in \textbf{Type}$, $\mathit{Arity}(P) = \langle T_1, \ldots, T_n \rangle$ ($\langle T_1, \ldots, T_n \rangle \in \mathit{Arity}(P)$) and $a_1 :_{\textbf{TYPE}_C} T_1, \ldots, a_n :_{\textbf{TYPE}_C} T_n$ then $P(a_1, \ldots a_n) \in \textbf{PType}$
6. $\textbf{PType} \subseteq \textbf{Type}$
7. for any $T \in \textbf{PType}$, $F(T)$ is a set disjoint from **Type**
8. for any $T \in \textbf{PType}$, $a :_{\textbf{TYPE}_C} T$ iff $a \in F(T)$

## *2.5 Complex types in record types

If we look back at the record type in (2) now we notice that there is something odd about the type constructed with the predicate temp_at_in, namely that the arguments to the predicate appear to be the *labels* 'e-time', 'e-location' and 'x' rather than *objects* that might occur under these labels in a record of this type. It is objects that are appropriate arguments to a predicate not the labels. (2) is actually a convenient abbreviatory notation for (3).

(3)
$$\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{e-time} & : & \mathit{Time} \\ \text{e-location} & : & \mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}} & : & \langle \lambda v_1{:}\mathit{Time}( \\ & & \quad \lambda v_2{:}\mathit{Loc}( \\ & & \quad\quad \lambda v_3{:}\mathit{Ind}( \\ & & \quad\quad\quad \text{temp\_at\_in}(v_1,v_2,v_3)))), \\ & & \quad\quad\quad \langle \text{e-time, e-location, x} \rangle \rangle \end{bmatrix}$$

Here what occurs in the $c_{temp\_at\_in}$-field is a pair whose first member is a function and whose second member is a list of labels indicating the fields in a record where the objects which are to be the arguments to the function are to be found. When applied to these objects the function will return a type constructed from the predicate and the objects.

For many simple cases such as (2) the abbreviatory notation is adequate and much easier to read as long as we keep in mind how it is to be interpreted. Care has to be taken, however, when record types are arguments to predicates. Consider a putative representation of a type corresponding to a reading of *some man appears to own a donkey* where *appear* is treated as corresponding to a one place predicate taking a record type as argument:

$$\left[\begin{array}{lcl} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \mathrm{man(x)} \\ & & \\ \mathrm{c}_3 & : & \mathrm{appear}\left(\left[\begin{array}{lcl} \mathrm{y} & : & \mathit{Ind} \\ \mathrm{c}_2 & : & \mathrm{donkey(y)} \\ \mathrm{c}_4 & : & \mathrm{own(x,y)} \end{array}\right]\right) \end{array}\right]$$

Technically, this notation is incorrect since 'x' occuring within the argument to 'appear' picks up a path outside of the record type in which it occurs. The full and correct notation for this type would be:

$$\left[\begin{array}{lcl} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \langle \lambda v\ \mathrm{man}(v), \langle \mathrm{x} \rangle \rangle \\ & & \\ \mathrm{c}_3 & : & \langle \lambda u\ \mathrm{appear}\left(\left[\begin{array}{lcl} \mathrm{y} & : & \mathit{Ind} \\ \mathrm{c}_2 & : & \langle \lambda v\ \mathrm{donkey}(v), \langle \mathrm{y} \rangle \rangle \\ \mathrm{c}_4 & : & \langle \lambda v\ \mathrm{own}(u,v), \langle \mathrm{y} \rangle \rangle \end{array}\right]\right), \langle \mathrm{x} \rangle \rangle \end{array}\right]$$

When labels are unique there is no harm in using the imprecise notation.

The full treatment of types constructed with predicates which depend on the values introduced in other fields as in these examples requires us to add functions and function types to our type theory. Furthermore, since the function returns a type, we will need a type of types since we want to be able to say that the function takes objects of types *Time*, *Loc* and *Ind* and returns a type, that is an object of type *Type*. Once we have done this we will be ready to give an explicit definition of record types.

## *2.6 Function types

A system of complex types $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle, \langle A, F \rangle \rangle$ *has function types* if

1. for any $T_1, T_2 \in \mathbf{Type}$, $(T_1 \to T_2) \in \mathbf{Type}$
2. for any $T_1, T_2 \in \mathbf{Type}$, $f :_{\mathbf{TYPE_C}} (T_1 \to T_2)$ iff $f$ is a function whose domain is $\{a \mid a :_{\mathbf{TYPE_C}} T_1\}$ and whose range is included in $\{a \mid a :_{\mathbf{TYPE_C}} T_2\}$

## *2.7 The type Type and stratification

An intensional type system is one in which the types themselves become objects of a type. We introduce a distinguished type *Type* to which all the members of the set **Type** belong. Things are a little more complicated than this, though, since we want *Type* itself to be a type and therefore it should belong to the set **Type**. This would mean that *Type* belongs to itself, i.e. *Type*:*Type*. Allowing types to belong to themselves puts us in danger of creating a situation in which Russell's paradox arises. If some members of **Type** belong to themselves then we should be able to talk of the set of types which do not belong to themselves, $\{T \in \mathbf{Type} \mid T \not: T\}$. Suppose that some model assigns this set to $T'$. Then the question arises whether $T'$ belongs to itself and we can show that if $T' : T'$ then $T' \not: T'$ and if $T' \not: T'$ then $T' : T'$.

In order to avoid this problem we will *stratify* (or *ramify*) our type system by introducing types of different *orders*. A type system of order 0 will be a system of complex types in the way we have defined it. The set of types, $\mathbf{Type}^1$ of a type system of order 1 based on this system will contain in addition to everything in the original type system a type, $Type^1$, to which all the types of order 0, members of the set $\mathbf{Type}^0$, belong. In general for all the natural numbers $n$, $Type^{n+1}$ will be a type to which all the types in $\mathbf{Type}^n$ belong. But there may be more additional types included in the higher sets of types. Suppose, for example, that we want to introduce a predicate $P$ expressing a relationship between individuals and types. (This will be our basic strategy for the treatment of attitude predicates such as *believe* and *know*.) Then $Arity(P)$ might be $\langle Ind, Type^n\rangle$. In systems of any order less than $n$, $P$ will not be able to be used to construct a type because clause 4 in our definition of systems of complex types requires that the types assigned to the arguments be types in the system. However, in systems of order $n$ or greater the required type will be present and the predicate will form a type.

This avoids the risk of running into Russell's paradox but it introduces another problem which it is best we deal with straight away. We will illustrate the problem by creating a small example. Suppose that we have a system of complex types which includes the type *Ind* ("individuals") to which the objects $a$, $b$ and $c$ belong. Suppose further that we have three predicates *run*,*know* and *believe* and that $Arity(run)$=$\langle Ind\rangle$ and $Arity(know)$=$Arity(believe)$=$\langle Ind, Type^1\rangle$. The set $\mathbf{Type}^0$ will contain the types $run(a)$, $run(b)$ and $run(c)$ but no types constructed with *know* and *believe*. The set $\mathbf{Type}^1$ will contain types such as $believe(a, run(a))$ and $know(c, run(b))$ in addition, since $run(a)$, $run(b)$ and $run(c)$, being members of $\mathbf{Type}^0$ will belong to the type $Type^1$. The set $\mathbf{Type}^2$ will not get any additional types constructed with predicates since the arity of the predicates restricts the second argument to be of $Type^1$. But suppose we want to express that $a$ believes that $b$ knows that $c$ runs, that is we want to construct the type $believe(a, know(b, run(c)))$. Perhaps we could solve this by saying that the arity of *know* and *believe* is $\langle Ind, Type^2\rangle$. But now $\mathbf{Type}^1$ will not contain any types constructed with these predicates and $\mathbf{Type}^2$ will again only contain types such as $know(c, run(b))$.

In order to solve this problem we need to introduce a limited amount of *polymorphism* into our arities and assign these predicates the arity $\langle Ind, Type^n\rangle_{n>0}$ (that is, the set of sequences $\langle Ind, Type^n\rangle$ where $n$ is a natural number greater than 0). Predicates with this arity will be able to take arguments of any type $Type^n$ where $n > 0$. We will say that the predicates *know* and *believe* have this arity. Now it will be the case that $run(c){:}Type^1$, $know(b, run(c)){:}Type^2$, $believe(a, know(b, run(c))){:}Type^3$ and so on.

An *intensional system of complex types* is a family of quadruples indexed by the natural numbers:

$$\mathbf{TYPE}_{IC} = \langle \mathbf{Type}^n, \mathbf{BType}, \langle \mathbf{PType}^n, \mathbf{Pred}, \mathbf{ArgIndices}, Arity\rangle, \langle A, F^n\rangle\rangle_{n\in Nat}$$

where (using $\mathbf{TYPE}_{IC_n}$ to refer to the quadruple indexed by $n$):

1. for each $n$,$\langle \mathbf{Type}^n, \mathbf{BType}, \langle \mathbf{PType}^n, \mathbf{Pred}, \mathbf{ArgIndices}, Arity\rangle, \langle A, F^n\rangle\rangle$ is a system of complex types
2. for each $n$, $\mathbf{Type}^n \subseteq \mathbf{Type}^{n+1}$ and $\mathbf{PType}^n \subseteq \mathbf{PType}^{n+1}$
3. for each $n$, if $T \in \mathbf{PType}^n$ and $p \in F^n(T)$ then $p \in F^{n+1}(T)$
4. for each $n > 0$, $Type^n \in \mathbf{Type}^n$
5. for each $n > 0$, $T :_{\mathbf{TYPE}_{IC_n}} Type^n$ iff $T \in \mathbf{Type}^{n-1}$

We can represent a stratified intensional system of types diagrammatically as Figure 2 where we represent just the first three levels of an infinite stratification.

An intensional system of complex types $\mathbf{TYPE}_{IC}$,

$$\mathbf{TYPE}_{IC} = \langle \mathbf{Type}^n, \mathbf{BType}, \langle \mathbf{PType}^n, \mathbf{Pred}, \mathbf{ArgIndices}, Arity\rangle, \langle A, F^n\rangle\rangle_{n\in Nat}$$

*has dependent function types* if

1. for any $n > 0$, $T \in \mathbf{Type}^n$ and $\mathcal{F} :_{\mathbf{TYPE}_{IC_n}} (T \rightarrow Type^n)$, $((a : T) \rightarrow \mathcal{F}(a)) \in \mathbf{Type}^n$
2. for each $n > 0$, $f :_{\mathbf{TYPE}_{IC_n}} ((a : T) \rightarrow \mathcal{F}(a))$ iff $f$ is a function whose domain is $\{a \mid a :_{\mathbf{TYPE}_{IC_n}} T\}$ and such that for any $a$ in the domain of $f$, $f(a) :_{\mathbf{TYPE}_{IC_n}} \mathcal{F}(a)$.

We might say that on this view dependent function types are "semi-intensional" in that they depend on there being a type of types for their definition but they do not introduce types as arguments to predicates and do not involve the definition of orders of types in terms of the types of the next lower order.

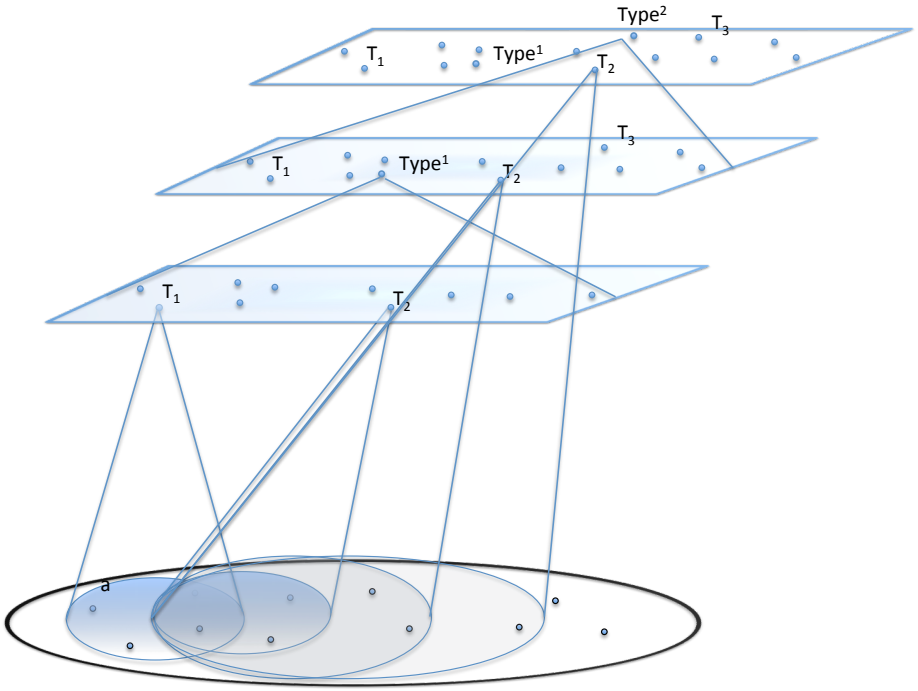


Figure 2. Intensional system of types with stratification

## *2.8 *Record types*

In this section we will define what it means for a system of complex types to have record types. The objects of record types, that is, records, are themselves structured mathematical objects of a particular kind and we will start by characterizing them.

A *record* is a finite set of ordered pairs (called *fields*) which is the graph of a function. If $r$ is a record and $\langle \ell, v \rangle$ is a field in $r$ we call $\ell$ a *label* and $v$ a *value* in $r$ and we use $r.\ell$ to denote $v$. $r.\ell$ is called a *path* in $r$.

We will use a tabular format to represent records. A record $\{\langle \ell_1, v_1 \rangle, \ldots, \langle \ell_n, v_n \rangle\}$ is displayed as

$$\begin{bmatrix} \ell_1 & = & v_1 \\ \ldots & & \\ \ell_n & = & v_n \end{bmatrix}$$

A value may itself be a record and paths may extend into embedded records. A

record which contains records as values is called a *complex record* and otherwise a record is *simple*. Values which are not records are called *leaves*. Consider a record $r$

$$\left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} \mathit{ff} & = & a \\ \mathit{gg} & = & b \end{array}\right] \\ g & = & c \end{array}\right] \\ g & = & \left[\begin{array}{lcl} h & = & \left[\begin{array}{lcl} g & = & a \\ h & = & d \end{array}\right] \end{array}\right] \end{array}\right]$$

Among the paths in $r$ are $r.f$, $r.g.h$ and $r.f.f.\mathit{ff}$ which denote, respectively,

$$\left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} \mathit{ff} & = & a \\ \mathit{gg} & = & b \end{array}\right] \\ g & = & c \end{array}\right]$$

$$\left[\begin{array}{lcl} g & = & a \\ h & = & d \end{array}\right]$$

and $a$. We will make a distinction between *absolute paths*, such as those we have already mentioned, which consist of a record followed by a series of labels connected by dots and *relative paths* which are just a series of labels connected by dots, e.g. $g.h$. Relative paths are useful when we wish to refer to similar paths in different records. We will use *path* to refer to either absolute or relative paths when it is clear from the context which is meant. The set of leaves of $r$, also known as its *extension* (those objects other than labels which it contains), is $\{a, b, c, d\}$. The bag (or multiset) of leaves of $r$, also known as its *multiset extension*, is $\{a, a, b, c, d\}$. A record may be regarded as a way of labelling and structuring its extension. Two records are *(multiset) extensionally equivalent* if they have the same (multiset) extension. Two important, though trivial, facts about records are:

> *Flattening.* For any record $r$, there is a multiset extensionally equivalent simple record. We can define an operation of flattening on records which will always produce an equivalent simple record. In the case of our example, the result of flattening is
>
> $$\left[\begin{array}{lcl} f.f.\mathit{ff} & = & a \\ f.f.\mathit{gg} & = & b \\ f.g & = & c \\ g.h.g & = & a \\ g.h.h & = & d \end{array}\right]$$
>
> assuming the flattening operation uses paths from the original record in a rather obvious way to create unique labels for the new record.
>
> *Relabelling.* For any record $r$, if $\pi_1.\ell.\pi_2$ is a path $\pi$ in $r$, and $\pi_1.\ell'.\pi_2{}'$ is *not* a path in $r$ (for any $\pi_2{}'$), then substituting $\ell'$ for the occurrence of

$\ell$ in $\pi$ results in a record which is multiset equivalent to $r$. We could, for example, substitute $k$ for the second occurrence of $g$ in the path $g.h.g$ in our example record.

$$\left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} f & = & \left[\begin{array}{lcl} ff & = & a \\ gg & = & b \end{array}\right] \\ g & = & c \end{array}\right] \\ g & = & \left[\begin{array}{lcl} h & = & \left[\begin{array}{lcl} k & = & a \\ h & = & d \end{array}\right] \end{array}\right] \end{array}\right]$$

A record *type* is a record in the general sense defined above where the values in its fields are types or, in some cases, certain kinds of mathematical objects which can be used to construct types.

A record $r$ is *well-typed* with respect to a system of types **TYPE** with set of types **Type** and a set of labels $L$ iff for each field $\langle \ell, a\rangle \in r$, $\ell \in L$ and either $a :_{\mathbf{TYPE}} T$ for some $T \in \mathbf{Type}$ or $a$ is itself a record which is well-typed with respect to **TYPE** and $L$.

A system of complex types $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F\rangle\rangle$ *has record types based on* $\langle L, \mathbf{RType}\rangle$, where $L$ is a countably infinite set (of labels) and $\mathbf{RType} \subseteq \mathbf{Type}$, where **RType** is defined by:

1. $\mathit{Rec} \in \mathbf{RType}$
2. $r :_{\mathbf{TYPE}_C} \mathit{Rec}$ iff $r$ is a well-typed record with respect to $\mathbf{TYPE}_C$ and $L$.
3. if $\ell \in L$ and $T \in \mathbf{Type}$, then $\{\langle \ell, T\rangle\} \in \mathbf{RType}$.
4. $r :_{\mathbf{TYPE}_C} \{\langle \ell, T\rangle\}$ iff $r :_{\mathbf{TYPE}_C} \mathit{Rec}$, $\langle \ell, a\rangle \in r$ and $a :_{\mathbf{TYPE}_C} T$.
5. if $R \in \mathbf{RType}$, $\ell \in L$, $\ell$ does not occur as a label in $R$ (i.e. there is no field $\langle \ell', T'\rangle$ in $R$ such that $\ell' = \ell$), then $R \cup \{\langle \ell, T\rangle\} \in \mathbf{RType}$.
6. $r :_{\mathbf{TYPE}_C} R \cup \{\langle \ell, T\rangle\}$ iff $r :_{\mathbf{TYPE}_C} R$, $\langle \ell, a\rangle \in r$ and $a :_{\mathbf{TYPE}_C} T$.

This gives us non-dependent record types in a system of complex types. We can extend this to intensional systems of complex types (with stratification).

An *intensional system of complex types* $\mathbf{TYPE}_{IC} = \langle \mathbf{Type}^n, \mathbf{BType}, \langle \mathbf{PType}^n, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F^n\rangle\rangle_{n\in Nat}$ *has record types based on* $\langle L, \mathbf{RType}^n\rangle_{n\in Nat}$ if for each $n$, $\langle \mathbf{Type}^n, \mathbf{BType}, \langle \mathbf{PType}^n, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F^n\rangle\rangle$ has record types based on $\langle L, \mathbf{RType}^n\rangle$ and

1. for each $n$, $\mathbf{RType}^n \subseteq \mathbf{RType}^{n+1}$
2. for each $n > 0$, $\mathit{RecType}^n \in \mathbf{RType}^n$
3. for each $n > 0$, $T :_{\mathbf{TYPE}_{IC_n}} \mathit{RecType}^n$ iff $T \in \mathbf{RType}^{n-1}$

Intensional type systems may in addition contain *dependent* record types.

An *intensional system of complex types* $\mathbf{TYPE}_{IC} = \langle \mathbf{Type}^n, \mathbf{BType}, \langle \mathbf{PType}^n, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F^n\rangle\rangle_{n\in Nat}$ *has dependent record types based on* $\langle L, \mathbf{RType}^n\rangle_{n\in Nat}$, if it has records types based on $\langle L, \mathbf{RType}^n\rangle_{n\in Nat}$ and for each $n > 0$

1. if $R$ is a member of $\mathbf{RType}^n$, $\ell \in L$ not occurring as a label in $R$, $T_1, \ldots, T_m \in \mathbf{Type}^n$, $R.\pi_1, \ldots, R.\pi_m$ are paths in $R$ and $\mathcal{F}$ is a function of type $((a_1 : T_1) \rightarrow \ldots \rightarrow ((a_m : T_m) \rightarrow \mathit{Type}^n)\ldots)$, then $R \cup \{\langle \ell, \langle \mathcal{F}, \langle \pi_1, \ldots, \pi_m\rangle\rangle\rangle\} \in \mathbf{RType}^n$.

2. $r :_{\mathbf{TYPE}_{IC_n}} R \cup \{\langle \ell, \langle \mathcal{F}, \langle \pi_1, \ldots, \pi_m\rangle\rangle\rangle\}$ iff $r :_{\mathbf{TYPE}_{IC_n}} R$, $\langle \ell, a\rangle$ is a field in $r$, $r.\pi_1 :_{\mathbf{TYPE}_{IC_n}} T_1, \ldots, r.\pi_m :_{\mathbf{TYPE}_{IC_n}} T_m$ and $a :_{\mathbf{TYPE}_{IC_n}} \mathcal{F}(r.\pi_1, \ldots, r.\pi_m)$.

We represent a record type $\{\langle \ell_1, T_1\rangle, \ldots, \langle \ell_n, T_n\rangle\}$ graphically as

$$\left[\begin{array}{lcl} \ell_1 & : & T_1 \\ \ldots & & \\ \ell_n & : & T_n \end{array}\right]$$

In the case of dependent record types we sometimes use a convenient notation representing e.g.

$$\langle \lambda u \lambda v\ \mathrm{love}(u, v), \langle \pi_1, \pi_2\rangle\rangle$$

as

$$\mathrm{love}(\pi_1, \pi_2)$$

Our systems now allow both function types and dependent record types and allow dependent record types to be arguments to functions. We have to be careful when considering what the result of applying a function to a dependent record type should be. Consider the following simple example:

$$\lambda v_0 :\mathit{RecType}(\left[\mathrm{c}_0{:}v_0\right])$$

What should be the result of applying this function to the record type

$$\left[\begin{array}{lcl} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \langle \lambda v_1 :\mathit{Ind}(\mathrm{dog}(v_1)), \langle \mathrm{x}\rangle\rangle \end{array}\right]$$

Given normal assumptions about function application the result would be

$$\left[\begin{array}{lcl} \mathrm{c}_0 & : & \left[\begin{array}{lcl} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \langle \lambda v_1 :\mathit{Ind}(\mathrm{dog}(v_1)), \langle \mathrm{x}\rangle\rangle \end{array}\right] \end{array}\right]$$

but this would be incorrect. In fact it is not a well-formed record type since x is not a path in it. Instead the result should be

$$\left[\begin{array}{lcl} \mathrm{c}_0 & : & \left[\begin{array}{lcl} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \langle \lambda v_1 :\mathit{Ind}(\mathrm{dog}(v_1)), \langle \mathrm{c}_0.\mathrm{x}\rangle\rangle \end{array}\right] \end{array}\right]$$

where the path from the top of the record type is specified. Note that this adjustment is only required when a record type is being substituted into a position that lies on a path within a resulting record type. It will not, for example, apply in a case where a record type is to be substituted for an argument to a predicate such as when applying the function

$$\lambda v_0 : \mathit{RecType}(\left[\mathrm{c}_0\text{:appear}(v_0)\right])$$

to

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \langle \lambda v : \mathit{Ind}(\mathrm{dog(v)}),\ \langle \mathrm{x} \rangle \rangle \\ \mathrm{c}_2 & : & \langle \lambda v : \mathit{Ind}(\mathrm{approach(v)}),\ \langle \mathrm{x} \rangle \rangle \end{bmatrix}$$

where the position of $v_0$ is in an "intensional context", that is, as the argument to a predicate and there is no path to this position in the record type resulting from applying the function. Here the result of the application is

$$\begin{bmatrix} \mathrm{c}_0 & : & \mathrm{appear}( \begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \langle \lambda v : \mathit{Ind}(\mathrm{dog(v)}),\ \langle \mathrm{x} \rangle \rangle \\ \mathrm{c}_2 & : & \langle \lambda v : \mathit{Ind}(\mathrm{approach(v)}),\ \langle \mathrm{x} \rangle \rangle \end{bmatrix} ) \end{bmatrix}$$

with no adjustment necessary to the paths representing the dependencies.[3] (Note that '$\mathrm{c}_0$.x' is not a path in this record type.)

These matters arise as a result of our choice of using paths to represent dependencies in record types (rather than, for example, introducing additional unique identifiers to keep track of the positions within a record type as has been suggested by Thierry Coquand). It seems like a matter of implementation rather than a matter of substance and it is straightforward to define a path-aware notion of substitution which can be used in the definition of what it means to apply a TTR function to an argument. If $f$ is a function represented by $\lambda v : T(\phi)$ and $\alpha$ is the representation of an object of type $T$, then the result of applying $f$ to $\alpha$, $f(\alpha)$, is represented by $\mathrm{Subst}(\alpha,v,\phi,\emptyset)$, that is, the result of substituting $\alpha$ for $v$ in $\phi$ with respect to the empty path where for arbitrary $\alpha, v, \phi, \pi$, $\mathrm{Subst}(\alpha,v,\phi,\pi)$ is defined as

1. extend-paths($\alpha$,$\pi$), if $\phi$ is $v$
2. $\phi$, if $\phi$ is of the form $\lambda v : T(\zeta)$, for some $T$ and $\zeta$ (i.e. don't do any substitution if $v$ is bound within $\phi$)
3. $\lambda u : T(\mathrm{Subst}(\alpha,v,\zeta,\pi))$, if $\phi$ is of the form $\lambda u : T(\zeta)$ and $u$ is not $v$.
4. $\begin{bmatrix} \ell_1 & : & \mathrm{Subst}(\alpha,v,T_1,\pi.\ell_1) \\ \dots & & \\ \ell_n & : & \mathrm{Subst}(\alpha,v,T_n,\pi.\ell_n) \end{bmatrix}$, if $\phi$ is $\begin{bmatrix} \ell_1 & : & T_1 \\ \dots & & \\ \ell_n & : & T_n \end{bmatrix}$

[3] This record corresponds to the interpretation of *it appears that a dog is approaching.*

5. $P$(Subst($\alpha$,$v$,$\beta_1$,$\pi$),...,Subst($\alpha$,$v$,$\beta_n$,$\pi$)), if $\alpha$ is $P(\beta_1, \ldots, \beta_n)$ for some predicate $P$

6. $\phi$ otherwise

extend-paths($\alpha$,$\pi$) is

1. $\langle f, \langle \pi.\pi_1, \ldots, \pi.\pi_n \rangle\rangle$, if $\alpha$ is $\langle f, \langle \pi_1, \ldots, \pi_n \rangle\rangle$

2. $\begin{bmatrix} \ell_1 & : & \text{extend-paths}(T_1, \pi) \\ \ldots & & \\ \ell_n & : & \text{extend-paths}(T_n, \pi) \end{bmatrix}$ if $\alpha$ is $\begin{bmatrix} \ell_1 & : & T_1 \\ \ldots & & \\ \ell_n & : & T_n \end{bmatrix}$

3. $P$(extend-paths($\beta_1, \pi$),...,extend-paths($\beta_n, \pi$)), if $\alpha$ is $P(\beta_1, \ldots, \beta_n)$ for some predicate $P$

4. $\alpha$, otherwise

## 3 FRAMES IN THE COMPOSITIONAL SEMANTICS OF VERBS

### *3.1 Verbs as functions from frames to frame types*

Consider an intransitive verb such as *run*. Basically, this corresponds to a predicate of individuals. Thus (4) would represent the type of events or situations where the individual Sam ('sam') runs.

(4) run(sam)

On FrameNet[4] *run* on one of its readings is associated with the frame `Self_motion`. Like many other frames in FrameNet this has a frame element `Time` which in this frame is explained in this case as "The time when the motion occurs". This is what Reichenbach [Reichenbach, 1947] called more generally *event time* and we will use the label 'e-time'. We will add an additional argument for a time to the predicate and create a frame-type (5).[5]

(5) $\begin{bmatrix} \text{e-time} & : & \mathit{TimeInt} \\ \text{c}_{\text{run}} & : & \text{run(sam,e-time)} \end{bmatrix}$

For the type (5) to be non-empty it is required that there be some time interval at which Sam runs. We use *TimeInt* as an abbreviation for the type of time intervals, (6).

[4] accessed 1st April, 2010

[5] Of course, we are ignoring many other frame elements which occur in FrameNet's `Self_motion` which could be added to obtain a more detailed semantic analysis.

$$(6)\quad \begin{bmatrix} \text{start} & : & \textit{Time} \\ \text{end} & : & \textit{Time} \\ \text{c} & : & \text{start}<\text{end} \end{bmatrix}$$

In (5) there are no constraints on the time interval apart from the requirement that Sam runs at that time. A record will be of this type just in case in provides some time interval at which Sam runs with the appropriate labels. Thus this frame type corresponds to a "tenseless proposition", something that is not available in the Priorean setup [Prior, 1957; Prior, 1967] that Montague employs where logical formulae without a tense operator correspond to a present tense interpretation. In order to be able to add tense to this we need to relate the event time to another time interval, normally the time which Reichenbach calls the speech time.[6] A past tense type anchored to a time interval $\iota$ is represented in (7).

$$(7)\quad \begin{bmatrix} \text{e-time} & : & \textit{TimeInt} \\ \text{c}_{\text{tns}} & : & \text{e-time.end}< \iota.\text{start} \end{bmatrix}$$

This requires that the end of the event time interval has to precede the start of the speech time interval. In order for a past-tense sentence *Sam ran* to be true we would need to find an object of both types (5) and (7). This is equivalent to requiring that there is an object in the result of merging the two types given in (8). (We make the notion of merge precise in section *3.2.)

$$(8)\quad \begin{bmatrix} \text{e-time} & : & \textit{TimeInt} \\ \text{c}_{\text{tns}} & : & \text{e-time.end}< \iota.\text{start} \\ \text{c}_{\text{run}} & : & \text{run(sam,e-time)} \end{bmatrix}$$

Suppose that we have an utterance $u$, that is, a speech event of type (9).

$$(9)\quad \begin{bmatrix} \text{phon} & : & \text{"sam"}^\frown\text{"ran"} \\ \text{s-time} & : & \textit{TimeInt} \\ \text{c}_{\text{utt}} & : & \text{uttered(phon,s-time)} \end{bmatrix}$$

where "sam"$^\frown$"ran" is the type of strings of an utterance of *Sam* concatenated with an utterance of *ran*. (See section *3.4 for a discussion of string types.) Then we can say that the speech time interval $\iota$ in (8) is $u$.s-time. That is, the past tense constraint requires that the event happened before the start of the speech event.

(8) is a type which is the content of an utterance of the sentence *Sam ran*. In order to obtain the content of the verb *ran* we need to create a function which abstracts over the first argument of the predicate. Because frames will play an

[6] Uses of historic present tense provide examples where the tense is anchored to a time other than the speech time.

important role as arguments to predicates below we will not abstract over individuals but rather over frames containing individuals. The content of the verb *ran* will be (10).

$$(10)\quad \lambda r{:}\big[\mathrm{x}{:}\mathit{Ind}\big]\Bigg(\begin{bmatrix} \text{e-time} & : & \mathit{TimeInt} \\ \mathrm{c_{tns}} & : & \text{e-time.end}{<}\,\iota.\text{start} \\ \mathrm{c_{run}} & : & \mathrm{run}(r.\mathrm{x},\text{e-time}) \end{bmatrix}\Bigg)$$

We show how this content can be utilized in a toy grammar in section *3.5.

## *3.2 Meets and merges

A system of complex types $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F\rangle\rangle$ *has meet types* if

1. for any $T_1, T_2 \in \mathbf{Type}$, $(T_1 \wedge T_2) \in \mathbf{Type}$
2. for any $T_1, T_2 \in \mathbf{Type}$, $a :_{\mathbf{TYPE_C}} (T_1 \wedge T_2)$ iff $a :_{\mathbf{TYPE_C}} T_1$ and $a :_{\mathbf{TYPE_C}} T_2$

This definition does not make precise exactly which mathematical object is denoted by $T_1 \wedge T_2$. Our intention is that it denote an object which contains the symbol '$\wedge$' as a component, for example, the triple $\langle \wedge, T_1, T_2\rangle$. Note that if $T_1$ and $T_2$ are record types as defined in section *2.8, then $T_1 \wedge T_2$ will *not* be a record type in the sense of this definition, since it is not a set of fields as required by the definition. This is true despite the fact that anything which is of the type $T_1 \wedge T_2$ where $T_1$ and $T_2$ are record types will be a record. There will, however, be a record type which is equivalent to the meet type.

There is a range of notions of equivalence which are available for types. For present purposes we will use a notion we call *necessary equivalence* which says that two types $T_1$ and $T_2$ are necessarily equivalent just in case $a : T_1$ iff $a : T_2$ on any assigment to basic types, $A$, and assignment to types constructed from a predicate and its arguments, $F$. This relates to the definition of a system of complex types in section *2.4, that is a system $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F\rangle\rangle$. The idea is that a notion of equivalence related to a single system of complex types $\mathbf{TYPE}_C$ that would say that $T_1$ is equivalent to $T_2$ just in case $a :_{\mathbf{TYPE_C}} T_1$ iff $a :_{\mathbf{TYPE_C}} T_2$ would be a weaker notion of "material equivalence". Necessary equivalence is a stronger notion that requires that $T_1$ and $T_2$ have the same extension no matter which functions $A$ and $F$ are chosen. We make this precise by introducing modal systems of complex types (section *3.3).

If $T_1$ and $T_2$ are record types then there will always be a record type (not a meet) $T_3$ which is necessarily equivalent to $T_1 \wedge T_2$. Let us consider some examples:

$$\big[\mathrm{f}{:}T_1\big] \wedge \big[\mathrm{g}{:}T_2\big] \approx \begin{bmatrix} \mathrm{f}{:}T_1 \\ \mathrm{g}{:}T_2 \end{bmatrix}$$

$$\big[\mathrm{f}{:}T_1\big] \wedge \big[\mathrm{f}{:}T_2\big] \approx \big[\mathrm{f}{:}T_1 \wedge T_2\big]$$

Below is a more logically oriented definition of the simplification of meets of record types than that given in [Cooper, 2008]. We define a function $\mu$ which maps meets

of record types to an equivalent record type, record types to equivalent types where meets in their values have been simplified by $\mu$ and any other types to themselves:

1. If for some $T_1$, $T_2$, $T = T_1 \wedge T_2$ then $\mu(T) = \mu'(\mu(T_1) \wedge \mu(T_2))$.
2. If $T$ is a record type then $\mu(T)$ is $T'$ such that for any $\ell$,$v$, $\langle \ell, \mu(v) \rangle \in T'$ iff $\langle \ell, v \rangle \in T$.
3. Otherwise $\mu(T) = T$.

$\mu'(T_1 \wedge T_2)$ is defined by:

1. if $T_1$ and $T_2$ are record types, then $\mu'(T_1 \wedge T_2) = T_3$ such that
   (a) for any $\ell, v_1, v_2$, if $\langle \ell, v_1 \rangle \in T_1$ and $\langle \ell, v_2 \rangle \in T_2$, then
      i. if $v_1$ and $v_2$ are $\langle \lambda u_1 : T'_1 \ldots \lambda u_i : T'_i(\phi), \langle \pi_1 \ldots \pi_i \rangle \rangle$ and $\langle \lambda u'_1 : T''_1 \ldots \lambda u'_k : T''_k(\psi), \langle \pi'_1 \ldots \pi'_k \rangle \rangle$ respectively, then $\langle \lambda u_1 : T'_1 \ldots \lambda u_i : T'_i, \lambda u'_1 : T''_1 \ldots \lambda u'_k : T''_k(\mu(\phi \wedge \psi)), \langle \pi_1 \ldots \pi_i, \pi'_1 \ldots \pi'_k \rangle \rangle \in T_3$
      ii. if $v_1$ is $\langle \lambda u_1 : T'_1 \ldots \lambda u_i : T'_i(\phi), \langle \pi_1 \ldots \pi_i \rangle \rangle$ and $v_2$ is a type (i.e. not of the form $\langle f, \Pi \rangle$ for some function $f$ and sequence of paths $\Pi$), then $\langle \lambda u_1 : T'_1 \ldots \lambda u_i : T'_i(\mu(\phi \wedge v_2)), \langle \pi_1 \ldots \pi_i \rangle \rangle \in T_3$
      iii. if $v_2$ is $\langle \lambda u'_1 : T''_1 \ldots \lambda u'_k : T''_k(\psi), \langle \pi'_1 \ldots \pi'_k \rangle \rangle$ and $v_1$ is a type, then $\langle \lambda u'_1 : T''_1 \ldots \lambda u'_k : T''_k(\mu(v_1 \wedge \psi)), \langle \pi'_1 \ldots \pi'_k \rangle \rangle \in T_3$
      iv. otherwise $\langle \ell, \mu(v_1 \wedge v_2) \rangle \in T_3$
   (b) for any $\ell, v_1$, if $\langle \ell, v_1 \rangle \in T_1$ and there is no $v_2$ such that $\langle \ell, v_2 \rangle \in T_2$, then $\langle \ell, v_1 \rangle \in T_3$
   (c) for any $\ell, v_2$, if $\langle \ell, v_2 \rangle \in T_2$ and there is no $v_1$ such that $\langle \ell, v_1 \rangle \in T_1$, then $\langle \ell, v_2 \rangle \in T_3$
2. Otherwise $\mu'(T_1 \wedge T_2) = T_1 \wedge T_2$

$T_1 \mathbin{\dot{\wedge}} T_2$ is used to represent $\mu(T_1 \wedge T_2)$.

This definition of $\mu$ differs from that given in [Cooper, 2008] in three respects. Firstly, it is not written in pseudocode and is therefore a better mathematical abstraction from the algorithm that has been implemented. Secondly, it includes the details of the treatment of dependencies within record types which were omitted from the previous definition. Finally, it excludes reference to a notion of subtype ('$\sqsubseteq$') which was included in the previous definition. This could be changed by adding the following clauses at the beginning of the definition of $\mu$ (after providing a characterization of the subtype relation, $\sqsubseteq$).

1. if for some $T_1, T_2$, $T = T_1 \wedge T_2$ and $T_1 \sqsubseteq T_2$ then $\mu(T) = T_1$
2. if for some $T_1, T_2$, $T = T_1 \wedge T_2$ and $T_2 \sqsubseteq T_1$ then $\mu(T) = T_2$

The current first clause would then hold in case neither of the conditions of these two clauses are met. The definition without these additional clauses only accounts for simplification of meets which have to do with merges of record types whereas the definition with the additional clauses would in addition have the effect, for example, that $\mu(T \wedge T_a) = T_a$ and $\mu(T_1 \wedge (T_1 \vee T_2)) = T_1$ (provided that we have an appropriate definition of $\sqsubseteq$) whereas the current definition without the additional clauses means that $\mu$ leaves these types unchanged.

## *3.3 Models and modal systems of types

Consider the definition of a system of complex types $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle, \langle A, F \rangle \rangle$ in section *2.4. We call the pair $\langle A, F \rangle$ a *model* because of its similarity to first order models. A model for classical first order logic provides a domain $A$ in which the logic is to be interpreted and an assignment $F$ of values based on $A$ to constants and predicates. That is: for any constant $c$, $F(c) \in A$; for a 1-place predicate $P$, $F(P) \subseteq A$; for a 2-place predicate $R$, $F(R) \subseteq A \times A$ and so on. Classical first order logic is not sorted, that is, $A$ is just a simple set of objects in the domain. Sorted first order logic provides a family of sets of different sorts. We can think of $A$ as a function which provides for each sort the objects which are of that sort. Predicates are then associated with an arity which tells us to which sort the arguments of the predicate should belong. Our models are similar to these models for sorted first order logic with our basic types corresponding to the sorts. Models of first order logic provide a way of making arbitrary connections between the basic expressions of the logic and another domain. Intuitively, we can think of the domain as being a part of the "real world" consisting of objects like people and tables, particularly if we are interested in natural language semantics. But domains can also be mathematical objects like numbers or sets. The exact nature of the objects in the domain in first order models is not of concern to the logician. It could, for example, be defined as a collection of sensor readings which are available to a particular robot. The model provides an interface between the logical expressions and some domain of our choosing. In a similar way the models in our type theory provide an interface between the type theory and a system external to the type theory of our choosing: the "real world", robot sensations or whatever.

The $F$ of our models behaves a little differently from that in first order models in that it assigns objects to types constructed from predicates rather than the predicates themselves. Suppose we have a type $P(a)$ constructed from a predicate $P$ and an object $a$ which is of an appropriate basic type as required by the arity of $P$. We could have made our models even closer to those of first order logic by having $F$ assign sets to predicates in the same way as in first order logic. We could mimic truth-values by introducing a distinguished basic type *Truth* such that $A(Truth) = \{true\}$. We could then have said that $true : P(a)$ iff $a \in F(P)$ and no other object $b$ is such that $b : P(a)$. From the perspective of type theory, however, this seems like an odd thing to do, in part because the object *true* seems

like an odd thing to have to include in your domain, being an artifical object which has to belong to so many different types and in part because it is missing a fundamental type theoretical intuition that something of type $P(a)$ is whatever it is that counts as a proof object for the fact that $a$ falls under the predicate $P$. So instead of having $F$ assign a value to the predicate $P$ we have it assign a value to the type $P(a)$. The exact nature of the object which is obtained by applying $F$ to $P(a)$ depends on what kind of model you are working with. A basic intuition corresponding to the idea that models represent the "real world" is that it is a situation (i.e. a "bit of the real world") which shows that $a$ falls under predicate $P$. But we could also define models in terms of robot sensations, four-dimensional space coordinates, databases, urls or whatever takes our fancy. The model is the place where we can connect our type theoretical system to some system external to the type theory. By moving from truth-values to this richer world of proof objects we have not lost the notion of truth. The "true" types are just those that are assigned a non-empty set of objects.

There is another important way in which our models are different from classical first order models. In first order logic the model relates two entirely separate domains: the syntactic expressions of first order logic and the model theoretic domain in which it is interpreted. The syntax of the logic is defined independently of the model. What counts as a well-formed expression does not depend in any way on what particular model we are using. This is not true of the models we have used in our type theoretical systems. Suppose that we have a predicate $P$ with arity $\langle T_1 \rangle$. Suppose furthermore that our model is such that $A(T_1) = \{a\}$ and $A(T_2) = \{b\}$. Then it will be the case that $P(a)$ is a type, but not $P(b)$. If we had chosen a different model the set of types might have been different. This fact alone might lead some people to conclude that it is confusing and misleading to call $\langle A, F \rangle$ a model in our type systems and certainly there is much of value in this point of view. The term 'model' is firmly entrenched in logic as that which provides the arbitrary parts of the interpretation of an independently defined syntactic language. However, we have chosen to persist in using the term here to emphasize the correspondence to models in logic and necessity of introducing an arbitrary link between a type theoretical system and an "external world" of some kind. The intuitive connection to models in logic is reinforced by our use of models in the discussion of modal systems below.

A modal system of complex types provides a collection of models, $\mathcal{M}$, so that we can talk about properties of the whole collection of type assignments provided by the various models $M \in \mathcal{M}$.

A *modal system of complex types based on* $\mathcal{M}$ is a family of quadruples:

$$\mathbf{TYPE}_{MC} = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle, M \rangle_{M \in \mathcal{M}}$$

where for each $M \in \mathcal{M}$, $\langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle, M \rangle$ is a system of complex types.

This enables us to define modal notions:

If $\mathbf{TYPE}_{MC} = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle, M \rangle_{M \in \mathcal{M}}$

is a modal system of complex types based on $\mathcal{M}$, we shall use the notation $\mathbf{TYPE}_{MC_M}$ (where $M \in \mathcal{M}$) to refer to that system of complex types in $\mathbf{TYPE}_{MC}$ whose model is $M$. Then:

1. for any $T_1, T_2 \in \mathbf{Type}$, $T_1$ *is (necessarily) equivalent to* $T_2$ *in* $\mathbf{TYPE}_{MC}$, $T_1 \approx_{\mathbf{TYPE}_{MC}} T_2$, iff for all $M \in \mathcal{M}$, $\{a \mid a :_{\mathbf{TYPE}_{MC_M}} T_1\} = \{a \mid a :_{\mathbf{TYPE}_{MC_M}} T_2\}$

2. for any $T_1, T_2 \in \mathbf{Type}$, $T_1$ *is a subtype of* $T_2$ *in* $\mathbf{TYPE}_{MC}$, $T_1 \sqsubseteq_{\mathbf{TYPE}_{MC}} T_2$, iff for all $M \in \mathcal{M}$, $\{a \mid a :_{\mathbf{TYPE}_{MC_M}} T_1\} \subseteq \{a \mid a :_{\mathbf{TYPE}_{MC_M}} T_2\}$

3. for any $T \in \mathbf{Type}$, $T$ *is necessary in* $\mathbf{TYPE}_{MC}$ iff for all $M \in \mathcal{M}$, $\{a \mid a :_{\mathbf{TYPE}_{MC_M}} T\} \neq \emptyset$

4. for any $T \in \mathbf{Type}$, $T$ *is possible in* $\mathbf{TYPE}_{MC}$ iff for some $M \in \mathcal{M}$, $\{a \mid a :_{\mathbf{TYPE}_{MC_M}} T\} \neq \emptyset$

## *3.4 *Strings and regular types*

A *string algebra over a set of objects* $O$ is a pair $\langle S, ^\frown \rangle$ where:

1. $S$ is the closure of $O \cup \{e\}$ ($e$ is the empty string) under the binary operation '$^\frown$' ("concatenation")

2. for any $s$ in $S$, $e^\frown s = s^\frown e = s$

3. for any $s_1, s_2, s_3$ in $S$, $(s_1^\frown s_2)^\frown s_3 = s_1^\frown (s_2^\frown s_3)$. For this reason we normally write $s_1^\frown s_2^\frown s_3$ or more simply $s_1 s_2 s_3$.

The objects in $S$ are called strings. Strings have length. $e$ has length 0, any object in $O$ has length 1. If $s$ is a string in $S$ with length $n$ and $a$ is an object in $O$ then $s^\frown a$ has length $n+1$. We use $s[n]$ to represent the $n$th element of string $s$.

We can define types whose elements are strings. Such types correspond to regular expressions and we will call them *regular types*. Here we will define just two kinds of such types: concatenation types and Kleene-+ types.

A system of complex types $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle, \langle A, F \rangle \rangle$ *has concatenation types* if

1. for any $T_1$, $T_2 \in \mathbf{Type}$, $T_1{}^\frown T_2 \in \mathbf{Type}$

2. $a : T_1{}^\frown T_2$ iff $a = x^\frown y$, $x : T_1$ and $y : T_2$

$\mathbf{TYPE}_C$ *has Kleene-+ types* if

1. for any $T \in \mathbf{Type}$, $T^+ \in \mathbf{Type}$

2. $a : T^+$ iff $a = x_1^\frown \ldots {}^\frown x_n$, $n > 0$ and for $i$, $1 \leq i \leq n$, $x_i : T$

Strings are used standardly in formal language theory where strings of symbols or strings of words are normally considered. Following important insights by Tim Fernando [Fernando, 2004; Fernando, 2006; Fernando, 2008; Fernando, 2009] we shall be concerned rather with strings of events. (We shall return to this in section 5.) We use informal notations like ‘ “sam” ’ and ‘ “ran” ’ to represent phonological types of speech events (utterances of *Sam* and *ran*). Thus ‘ “sam”⌢“ran” ’ is the type of speech events which are concatenations of an utterance of *Sam* and an utterance of *ran*.

## *3.5 Grammar and compositional semantics

In order to illustrate how the content we have given for *ran* in section 3.1 figures in a grammar and compositional semantics we shall define a toy grammar which covers the sentence *Sam ran*.

We will present our grammar in terms of signs (in a similar sense to HPSG, see, for example, [Sag *et al.*, 2003]). Our signs will be records of type *Sign* which for present purposes we will take to be the type:

$$\left[\begin{array}{lcl} \text{s-event} & : & \mathit{SEvent} \\ \text{synsem} & : & \left[\begin{array}{lcl} \text{cat} & : & \mathit{Cat} \\ \text{cnt} & : & \mathit{Cnt} \end{array}\right] \end{array}\right]$$

We shall spell out the nature of the types *SEvent*, *Cat* and *Cnt* below. A sign has two main components, one corresponding to the physical nature of the speech event (‘s-event’) and the other to its interpretation (syntax and semantics, ‘synsem’, using the label which is well-established in HPSG).

*SEvent* is the type

$$\left[\begin{array}{lcl} \text{phon} & : & \mathit{Phon} \\ \text{s-time} & : & \left[\begin{array}{lcl} \text{start} & : & \mathit{Time} \\ \text{end} & : & \mathit{Time} \end{array}\right] \\ \text{utt}_{\text{at}} & : & \langle \lambda v_1 : \mathit{Str}(\lambda v_2 : \mathit{Time}(\lambda v_3 : \mathit{Time}(\text{uttered\_at}(v_1,v_2,v_3)))), \\ & & \quad \langle \text{s-event.phon, s-event.s-time.start, s-event.s-time.end} \rangle \rangle \end{array}\right]$$

In the s-event component the phon-field represents the phonology of an expression. Here we will take phonology as a string of word utterances although in a complete treatment of spoken language we would need phonological and phonetic attributes. That is we take *Phon* to be $\mathit{Wrd}^+$ where *Wrd* (the type of word utterances) is defined in the lexicon. The s-time (“speech time”) field represents the starting and ending time for the utterance. We assume the existence of a predicate ‘uttered_at’ with arity $\langle \mathit{Phon}, \mathit{Time}, \mathit{Time} \rangle$. An object of type ‘uttered_at($a$,$t_1$,$t_2$)’ could be an event where $a$ is uttered beginning at $t_1$ and ending at $t_2$ or a corresponding hypothesis produced by a speech recognizer with time-stamps, depending on the application of the theory. In a more complete treatment we would need additional information about the physical nature of the speech event, such as the identity of the speaker and where it took place.

In the synsem component the cat-field introduces a category for the phrase. For present purposes we will require that the following hold of the type *Cat*:

s, np, vp, $\mathrm{n}_{\mathrm{prop}}$, $\mathrm{v}_{\mathrm{i}}$ : *Cat*

The objects of type *Cat* (s, np, vp etc.) are regarded as convenient abstract objects which are used to categorize classes of speech events.

The cnt-field represents the content or interpretation of the utterance. Since the content types become rather long we will introduce abbreviations to make them readable:

*Ppty*, "property" is to be $\left[\mathrm{x}{:}\mathit{Ind}\right]{\rightarrow}\mathit{RecType}$
*Quant*, "quantifier" is to be $\mathit{Ppty}{\rightarrow}\mathit{RecType}$

We only use a small finite number of function types for content types and thus we are able to define the type *Cnt* for present purposes as

$\mathit{RecType}\vee(\mathit{Ppty}\vee\mathit{Quant})$

This makes use of join types which are defined in a similar way to meet types: $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F\rangle\rangle$ *has join types* if

1. for any $T_1, T_2 \in \mathbf{Type}$, $(T_1 \vee T_2) \in \mathbf{Type}$
2. for any $T_1, T_2 \in \mathbf{Type}$, $a :_{\mathbf{TYPE_C}} (T_1 \vee T_2)$ iff $a :_{\mathbf{TYPE_C}} T_1$ or $a :_{\mathbf{TYPE_C}} T_2$

We will present first the lexicon and then rules for combining phrases.

## *Lexicon*

We will define lexical functions which tell us how to construct a type for a lexical item on the basis of a phonological type and either an object or a type corresponding to an observation of the world. The idea is that an agent which is constructing a grammar for use in a particular communicative situation will construct lexical types on the basis of a coordinated pair of observations: an observation of a speech event and an observation of an object or event with which the speech event is associated. This is related to the idea from situation semantics that meaning is a relation between an utterance situation and a described situation [Barwise and Perry, 1983]. The use of types here relates to the idea of type judgements as being involved in perception as discussed in section *2.3.

We shall use the following notation:

> If $W$ is a phonological type, then $\mathrm{c}_W$ is a distinguished label associated with $W$, such that if $W_1 \neq W_2$ then $\mathrm{c}_{W_1} {\neq} \mathrm{c}_{W_2}$.

We shall also make use of singleton types. $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity}\rangle, \langle A, F\rangle\rangle$ *has singleton types* if

1. for any $T \in \mathbf{Type}$ and $b :_{\mathbf{TYPE_C}} T$, $T_b \in \mathbf{Type}$
2. for any $T \in \mathbf{Type}$, $a :_{\mathbf{TYPE_C}} T_b$ iff $a :_{\mathbf{TYPE_C}} T$ and $a = b$

In the case of a singleton type $T_x$ we allow a variant notation in records (corresponding to the manifest fields of Coquand et al., 2004) using

$$\left[\ \ell{=}x \quad : \quad T\ \right]$$

for

$$\left[\ \ell \quad : \quad T_x\ \right]$$

When we have a field

$$\left[\ \ell \quad : \quad \langle \lambda v_1 : T_1 \ldots \lambda v_n : T_n(T_x), \langle \pi_1 \ldots \pi_n \rangle \rangle\ \right]$$

we allow for convenience notations such as

$$\left[\ \ell{=}\langle \lambda v_1 : T_1 \ldots \lambda v_n : T_n\mathbf{\{}x\mathbf{\}}, \langle \pi_1 \ldots \pi_n \rangle \rangle \quad : \quad T\ \right]$$
$$\left[\ \ell{=}x \quad : \quad \langle \lambda v_1 : T_1 \ldots \lambda v_n : T_n(T), \langle \pi_1 \ldots \pi_n \rangle \rangle\ \right]$$

or

$$\left[\ell{=}\langle \lambda v_1 : T_1 \ldots \lambda v_n : T_n\mathbf{\{}x\mathbf{\}}, \langle \pi_1 \ldots \pi_n \rangle \rangle{:}\langle \lambda v_1 : T_1 \ldots \lambda v_n : T_n(T), \langle \pi_1 \ldots \pi_n \rangle \rangle\right]$$

depending on how $T_x$ depends on $\pi_1 \ldots \pi_n$. We use $\mathbf{\{}$ and $\mathbf{\}}$ to delimit $x$ since $x$ itself may be a function thus leading to ambiguity in the notation if we do not distinguish which $\lambda$'s represent dependency and which belong to the resulting object. Note that this ambiguity only arises in the notation we are adopting for convenience.

*Proper names*

The most straightforward view of proper names is that they are based on pairings of proper noun utterances and individuals. While the full story about proper names may have to be more complex, this will suffice for our present purposes.

We define a function $\mathrm{lex}_{\mathrm{n_{Prop}}}$ which maps phonological types corresponding to proper names like *Sam* and individuals to record types, such that if $W$ is a phonological type such as "Sam" or "John" and $a$:*Ind*, $\mathrm{lex}_{\mathrm{n_{Prop}}}(W, a)$ is

$$\mathit{Sign} \wedge \left[\begin{array}{lcl} \text{s-event} & : & \left[\ \text{phon} \quad : \quad W\ \right] \\ \text{synsem} & : & \left[\begin{array}{lcl} \text{cat}{=}\mathrm{n_{Prop}} & : & \mathit{Cat} \\ \text{cnt}{=}\lambda v{:}\mathit{Ppty}(v(\left[\text{x}{=}a\right])) & : & \mathit{Quant} \end{array}\right] \end{array}\right]$$

The idea of this function is that an agent could have it as a resource to construct a lexical item for a local language on observing a pairing of a particular type of utterance (e.g. utterances of *Sam*) and a particular individual. If the language we

are building is small enough there will be only one individual associated with a given phonological type such as "sam" but it is easy to imagine situations where there will be a need to have different individuals associated with the same name even within a local language, for example, if you need to talk about two people named *Sam* who write a book together. While this creates potential for misunderstanding there is nothing technically mysterious about having two lexical types which happen to share the same phonology. This is in contrast to the classical formal semantics view of proper names as related to logical constants where it seems unexpected that proper nouns should be able to refer to different individuals on different uses.

An example of a set of basic proper names which could be generated with these resources given two individuals $a$ and $b$ (that is, $a, b{:}Ind$) would be

$$\{\mathrm{lex}_{\mathrm{n_{Prop}}}(\text{“Sam”},a),\\ \mathrm{lex}_{\mathrm{n_{Prop}}}(\text{“John”},b)\}$$

*Intransitive verbs*

For intransitive verbs we will take the paired observations to involve a phonological type corresponding to an intransitive verb on the one hand and a predicate on the other. Philosophically, it may appear harder to explain what it means to observe a predicate compared to observing an individual, even though if you dig deep enough even individuals are problematical. However, it seems that any reasonable theory of perception should account for the fact that we perceive the world in terms of various kinds of objects standing in relations to each other. Our predicates correspond to these relations and we would want to say that our cognitive apparatus is such that relations are reified in a way that they need to be in order to become associated with types of utterances. For a verb like *run* we will say that the predicate is one that holds between individuals and time intervals. We will argue in section 4 that for other verbs we need frames instead of individuals.

We define a function $\mathrm{lex}_{\mathrm{V_i}}$ which maps phonological types corresponding to intransitive verbs like *run* and predicates with arity $\langle Ind, TimeInt\rangle$, such that if $W$ is a phonological type like "run" or "walk" and $p$ is a predicate with arity $\langle Ind, TimeInt\rangle$, $\mathrm{lex}_{\mathrm{V_i}}(W, p)$ is

$$Sign \wedge \begin{bmatrix} \text{s-event}{:}\begin{bmatrix}\text{phon}{:}W\end{bmatrix} \\ \text{synsem}{:}\begin{bmatrix} \text{cat}{=}\mathrm{v_i}{:}Cat \\ \text{cnt}{=}\lambda r{:}\begin{bmatrix}\text{x}{:}Ind\end{bmatrix}\ (\begin{bmatrix}\text{e-time}{:}TimeInt \\ \mathrm{c}_W{:}\langle\lambda v{:}TimeInt(p(r.\text{x},v),\ \langle\text{e-time}\rangle\rangle\end{bmatrix}))){:}Ppty\end{bmatrix}\end{bmatrix}$$

Similar remarks hold for this function as for the one we used for proper names. For different local languages different predicates may be associated with utterances of *run* and even within the same local language, confusing though it may be, we may need to associate different predicates with different occurrences of *run*. In

this way verbs are like proper names and one can think of verbs as proper names of predicates.

However, this is not quite enough if we want to handle different forms of verbs such as infinitives, and present and past tenses. For purposes of simplification as our concern is not with the details of morphological types we will assume that all finite verb occurrences are third person singular and will not represent these features. In order to achieve this we need to define $\text{lex}_{\text{V}_\text{i}}$ not in terms of a single phonological type but a paradigm of phonological types corresponding to different configurations of morphological features. For present purposes we will think of there just being one morphological feature of tense which can take the values: inf ("infinitive"), pres ("present tense"), past ("past tense"). We will think of paradigms as functions which map records of type $\left[\text{tns}: \mathit{Tns}\right]$ to phonological types. Here the type *Tns* has elements inf, pres and past. Let **run** be the paradigm for *run*. The function is defined by

$\mathbf{run}(\left[\text{tns=inf}\right])=$ "run"
$\mathbf{run}(\left[\text{tns=pres}\right])=$ "runs"
$\mathbf{run}(\left[\text{tns=past}\right])=$ "ran"

and for *walk* we have

$\mathbf{walk}(\left[\text{tns=inf}\right])=$ "walk"
$\mathbf{walk}(\left[\text{tns=pres}\right])=$ "walks"
$\mathbf{walk}(\left[\text{tns=past}\right])=$ "walked"

In order to obtain the interpretations of the tensed forms of the verb we will need the following functions for present and past tense.

**Pres** which is to be $\lambda t{:}\mathit{TimeInt}(\begin{bmatrix}\text{e-time}{:}\mathit{TimeInt}\\ \text{tns}{:}\langle\lambda v{:}\mathit{TimeInt}(v = t),\ \langle\text{e-time}\rangle\rangle\end{bmatrix})$

**Past** which is to be $\lambda t{:}\mathit{TimeInt}(\begin{bmatrix}\text{e-time}{:}\mathit{TimeInt}\\ \text{tns}{:}\langle\lambda v{:}\mathit{TimeInt}(v.\text{end}< t.\text{start}),\langle\text{e-time}\rangle\rangle\end{bmatrix})$

The present tense function expresses that the event time is identical with the interval to which it is being compared. This is normally the speech time as in the grammar defined here, though it could also be a different time interval, for example in the interpretation of historic presents. The past tense function expresses that the end of the event time interval has to be prior to the start of the interval (e.g. the speech time) with which it is being compared.

We need also to make the distinction between finite and non-finite verb utterances and we will do this by introducing a field labelled 'fin' which will take values in the type *Bool* ("boolean") whose members are 0 and 1.

Now we redefine $\text{lex}_{\text{V}_\text{i}}$ to be a function which takes a paradigm $\mathcal{W}$ such as **run** or **walk**, a predicate $p$ with arity $\langle \mathit{Ind}, \mathit{TimeInt}\rangle$ and morphological record $m$ of type $\left[\text{tns}: \mathit{Tns}\right]$ such that

1. if $m$ is $\left[\text{tns=inf}\right]$, $\text{lex}_{\text{V}_\text{i}}(\mathcal{W}, p, m)$ is

$$\textit{Sign} \wedge \begin{bmatrix} \text{s-event:}\left[\text{phon:}\mathcal{W}(m)\right] \\ \text{synsem:} \begin{bmatrix} \text{cat=v}_\text{i}\text{:}\textit{Cat} \\ \text{fin=0:}\textit{Bool} \\ \text{cnt=}\lambda r\text{:}\left[\text{x:}\textit{Ind}\right]\ (\begin{bmatrix}\text{e-time:}\textit{TimeInt} \\ \text{c}_{\mathcal{W}(m)}\text{:}\langle\lambda v\text{:}\textit{TimeInt}(p(r.\text{x},v)),\ \langle\text{e-time}\rangle\rangle\end{bmatrix}\text{):}\textit{Ppty} \end{bmatrix} \end{bmatrix}$$

2. if $m$ is $\left[\text{tns=pres}\right]$, $\text{lex}_{\text{V}_\text{i}}(\mathcal{W}, p, m)$ is

$$\textit{Sign} \wedge \begin{bmatrix} \text{s-event:}\begin{bmatrix}\text{phon:}\mathcal{W}(m) \\ \text{s-time:}\textit{TimeInt}\end{bmatrix} \\ \text{synsem:} \begin{bmatrix} \text{cat=v}_\text{i}\text{:}\textit{Cat} \\ \text{fin=1:}\textit{Bool} \\ \text{cnt=}\langle\lambda v_1\text{:}\textit{Time}\{\lambda r\text{:}\left[\text{x:}\textit{Ind}\right] \\ \qquad (\begin{bmatrix}\text{e-time:}\textit{TimeInt} \\ \text{c}_{\mathcal{W}(m)}\text{:}\langle\lambda v_2\text{:}\textit{TimeInt}(p(r.\text{x},v_2)),\ \langle\text{e-time}\rangle\rangle\end{bmatrix}\wedge\mathbf{Pres}(v_1))\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \langle\text{s-event.s-time}\rangle\rangle\text{:}\textit{Ppty} \end{bmatrix} \end{bmatrix}$$

3. if $m$ is $\left[\text{tns=past}\right]$, $\text{lex}_{\text{V}_\text{i}}(\mathcal{W}, p, m)$ is

$$\textit{Sign} \wedge \begin{bmatrix} \text{s-event:}\begin{bmatrix}\text{phon:}\mathcal{W}(m) \\ \text{s-time:}\textit{TimeInt}\end{bmatrix} \\ \text{synsem:} \begin{bmatrix} \text{cat=v}_\text{i}\text{:}\textit{Cat} \\ \text{fin=1:}\textit{Bool} \\ \text{cnt=}\langle\lambda v_1\text{:}\textit{Time}\{\lambda r\text{:}\left[\text{x:}\textit{Ind}\right] \\ \qquad (\begin{bmatrix}\text{e-time:}\textit{TimeInt} \\ \text{c}_{\mathcal{W}(m)}\text{:}\langle\lambda v_2\text{:}\textit{TimeInt}(p(r.\text{x},v_2)),\ \langle\text{e-time}\rangle\rangle\end{bmatrix}\wedge\mathbf{Past}(v_1))\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \langle\text{s-event.s-time}\rangle\rangle\text{:}\textit{Ppty} \end{bmatrix} \end{bmatrix}$$

An example of a set of intransitive verbs which could be generated with these resources given appropriate predicates 'run' and 'walk' is

$$\bigcup_{\alpha\in\{\text{inf,pres,past}\}}\{\text{lex}_{\text{V}_\text{i}}(\textbf{run},\text{run},\left[\text{tns=}\alpha\right]), \\ \text{lex}_{\text{V}_\text{i}}(\textbf{walk},\text{walk},\left[\text{tns=}\alpha\right])\}$$

## *Syntactic and semantic composition*

We will think of composition rules as functions which take a string of utterances of various types and return a type for the whole string. That is, the basic form of our composition rules will be:

$$\lambda s : T_1(T_2)$$

where $T_1$ is a type of strings of signs and $T_2$ is a type of signs. More specifically we can say that *unary* rules are functions of the form

$\lambda s : T_1(T_2)$, where $T_1, T_2 \sqsubseteq Sign$

and *binary* rules are of the form

$\lambda s : T_1^\frown T_2(T_3)$, where $T_1, T_2, T_3 \sqsubseteq Sign$

'$\sqsubseteq$' here denotes the subtype relation defined in section *3.3. (We are suppressing the subscript used there.) We can, of course, generalize these notions to $n$-ary rules but unary and binary will be sufficient for our present purposes.

Note that to say that there is a string of signs $s_1^\frown s_2$ does not necessarily mean that the signs are temporally ordered in the sense that $s_1$.s-event.s-time.end $<$ $s_2$.s-event.s-time.start. There could be an advantage in this for the treatment of discontinuous constituents or free word order. But we can also define a special "temporal concatenation" type for concatenation of signs:

> A system of complex types $\mathbf{TYPE}_C = \langle \mathbf{Type}, \mathbf{BType}, \langle \mathbf{PType}, \mathbf{Pred}, \mathbf{ArgIndices}, \mathit{Arity} \rangle, \langle A, F \rangle \rangle$ *has temporal concatenation types for the type Sign* if
>
> 1. for any $T_1$, $T_2 \sqsubseteq$ *Sign*, $T_1{}^\frown_{\text{temp}} T_2 \in \mathbf{Type}$
> 2. $s : T_1{}^\frown_{\text{temp}} T_2$ iff $s = s_1^\frown s_2$, $s_1 : T_1$, $s_2 : T_2$ and $s_1$.s-event.s-time.end $<$ $s_2$.s-event.s-time.start.

We will factor our rules into component functions which we will then combine in order to make a complete rule. The components we will use here are:

**unary_sign** which we define to be

> $\lambda s{:}\mathit{Sign}(\mathit{Sign})$
>
> This takes any sign and returns the type *Sign*

**binary_sign** which we define to be

> $\lambda s{:}\mathit{Sign}^\frown_{\text{temp}}\mathit{Sign}(\mathit{Sign})$
>
> This takes any temporal concatenation of two signs and returns the type *Sign*

**phon_id** which we define to be

> $\lambda s{:}\big[\text{s-event:}\big[\text{phon:}\mathit{Phon}\big]\big]\big(\big[\text{s-event:}\big[\text{phon=}s\text{.s-event.phon:}\mathit{Phon}\big]\big]\big)$
>
> This takes any record $s$ of type $\big[\text{s-event:}\big[\text{phon:}\mathit{Phon}\big]\big]$ and returns a type which is the same except that the phonology field is now required to be filled by the value of that field in $s$.

**phon_concat** which we define to be

$$\lambda s{:}\big[\text{s-event}{:}\big[\text{phon}{:}\mathit{Phon}\big]\big]^\frown\big[\text{s-event}{:}\big[\text{phon}{:}\mathit{Phon}\big]\big]$$
$$\quad(\big[\text{s-event}{:}\big[\text{phon}{=}s[1].\text{s-event.phon}^\frown s[2].\text{s-event.phon}{:}\mathit{Phon}\big]\big])$$

This takes a string of two records with phonology fields and returns the type of a single record with a phonology field whose value is required to be the concatenation of the values of the phonology fields in the first and second elements of the string.

**unary_cat** which we define to be

$$\lambda c_1{:}\mathit{Cat}(\lambda c_2{:}\mathit{Cat}(\lambda s{:}\big[\text{cat}{=}c_1{:}\mathit{Cat}\big](\big[\text{cat}{=}c_2{:}\mathit{Cat}\big])))$$

This takes two categories and returns a function which maps a record with a category field with value the first category to a type of records with a category field which is required to be filled by the second category.

**binary_cat** which we define to be

$$\lambda c_1{:}\mathit{Cat}(\lambda c_2{:}\mathit{Cat}(\lambda c_3{:}\mathit{Cat}(\lambda s{:}\big[\text{cat}{=}c_1{:}\mathit{Cat}\big]^\frown\big[\text{cat}{=}c_2{:}\mathit{Cat}\big](\big[\text{cat}{=}c_3{:}\mathit{Cat}\big])))$$

This takes three categories and returns a function which maps a string of two records with a category field with values identical to the respective categories to a type of records with a category field which is required to be filled by the third category.

**cnt_id** which we define to be

$$\lambda s{:}\big[\text{synsem}{:}\big[\text{cnt}{:}\mathit{Cnt}\big]\big](\big[\text{synsem}{:}\big[\text{cnt}{=}s.\text{synsem.cnt}{:}\mathit{Cnt}\big]\big])$$

This takes any record $s$ of type $\big[\text{synsem}{:}\big[\text{cnt}{:}\mathit{Cnt}\big]\big]$ and returns a type which is the same except that the content field is now required to be filled by the value of that field in $s$.

**cnt_forw_app** which we define to be

$$\lambda T_1{:}\mathit{Type}(\lambda T_2{:}\mathit{Type}(\lambda s{:}\big[\text{synsem}{:}\big[\text{cnt}{:}T_1 \to T_2\big]\big]^\frown\big[\text{synsem}{:}\big[\text{cnt}{:}T_1\big]\big]$$
$$\quad(\big[\text{synsem}{:}\big[\text{cnt}{=}s[1].\text{synsem.cnt}(s[2].\text{synsem.cnt}){:}T_2\big]\big])$$

This takes any binary string of records $s$ such that the content of the first record is a function which takes arguments of a type to which the content of the second record belongs and returns a type whose content field is now required to be filled by the result of applying the content of the first record to the content of the second record.

**fin_id** which we define to be

$$\lambda s{:}\big[\text{fin}{:}\mathit{Bool}\big](\big[\text{fin}{=}s.\text{fin}{:}\mathit{Bool}\big])$$

This requires that the value of a 'fin'-field will be copied into the new type (corresponding to feature percolation in a non-branching tree in a more traditional feature-based grammar).

**fin_hd** which we define to be

$$\lambda s{:}Sign^\frown\big[\text{fin}{=}1{:}Bool\big]\big(\big[\text{fin}{=}s.\text{fin}{:}Bool\big]\big)$$

This requires that the second sign in a string of two has a positive specification for finiteness and copies it into the new type.

We will use the notion of merge defined in section *3.2 in the characterization of how these component functions are to be combined in order to form rules. Since the combination of these functions is so closely connected to the merge operation we will use a related symbol '$\wedge\!\!:$' with two dots rather than one. In the following definition we will use $T_i$ to represent types which are not string types and $v$ to represent an arbitrary variable.

1. $\lambda v{:}T_1(T_2) \wedge\!\!: \lambda v{:}T_3(T_4)$ is to be $\lambda v{:}T_1 \wedge\!\!: T_3(T_2 \wedge\!\!: T_4)$
2. $\lambda v{:}T_1^\frown T_2(T_3) \wedge\!\!: \lambda v{:}T_4^\frown T_5(T_6)$ is to be $\lambda v{:}(T_1 \wedge\!\!: T_4)^\frown(T_2 \wedge\!\!: T_5)\ (T_3 \wedge\!\!: T_6)$
3. $\lambda v{:}T_1^{\frown_{temp}} T_2(T_3) \wedge\!\!: \lambda v{:}T_4^\frown T_5(T_6)$ is to be $\lambda v{:}(T_1 \wedge\!\!: T_4)^{\frown_{temp}}(T_2 \wedge\!\!: T_5)\ (T_3 \wedge\!\!: T_6)$

Since $\wedge\!\!:$, like $\wedge$, is associative we will write $f \wedge\!\!: g \wedge\!\!: h$ instead of $(f \wedge\!\!: g) \wedge\!\!: h$ or $f \wedge\!\!: (g \wedge\!\!: h)$.

Now we can use the rule components we have defined to express the three rules we need for this small fragment.

**S → NP VP**

$$\begin{aligned}&\textsf{binary\_sign} \wedge\!\!: \textsf{phon\_concat} \wedge\!\!: \textsf{binary\_cat}(\text{np})(\text{vp})(\text{s}) \wedge\!\!: \textsf{fin\_hd}\\ &\qquad\qquad \wedge\!\!: \textsf{cnt\_forw\_app}(Ppty)(RecType)\end{aligned}$$

**NP → N**

$$\textsf{unary\_sign} \wedge\!\!: \textsf{phon\_id} \wedge\!\!: \textsf{unary\_cat}(\text{n}_{\text{Prop}})(\text{np}) \wedge\!\!: \textsf{cnt\_id}$$

**VP → $\mathsf{V}_i$**

$$\textsf{unary\_sign} \wedge\!\!: \textsf{phon\_id} \wedge\!\!: \textsf{unary\_cat}(\text{v}_i)(\text{vp}) \wedge\!\!: \textsf{fin\_id} \wedge\!\!: \textsf{cnt\_id}$$

This gives us a concise way to express rather complex functions corresponding to simple rules. The point of this is, however, not merely to give us yet another formalism for expressing natural language phrase structure and its interpretation but to show how such rules can be broken down into abstract components which an agent learning the language could combine in order to create rules which it has not previously had available in its resources. Thus an agent (such as a child in the one-word stage) which does not have a rule S → NP VP but who observes strings of linguistic events where NP's are followed by VP's may reason its way to a rule that combine NP-events followed by VP-events into a single event. While this concerns linguistic events it is closely related to the way we take strings of

non-linguistic events to form single events, for example, a going-to-bed-event for a child might normally consist of a string of events having-hot-milk⌢putting-on-pyjamas⌢getting-into-bed⌢listening-to-a-story. Our general ability to perceive events, that is, assign types to events and to combine these types into larger event types seems to be a large part of the basis for our linguistic ability. We will return to this in our discussion of Fernando's string theory of events in section 5.

## 4 USING FRAMES TO SOLVE A CLASSICAL PUZZLE ABOUT TEMPERATURE AND PRICES

### *4.1 The Partee puzzle*

Montague [1973] introduces a puzzle presented to him by Barbara Partee:

> From the premises **the temperature is ninety** and **the temperature rises**, the conclusion **ninety rises** would appear to follow by normal principles of logic; yet there are occasions on which both premises are true, but none on which the conclusion is.

Exactly similar remarks can be made substituting *price* for *temperature*. Montague's [1973] solution to this puzzle was to analyze *temperature*, *price* and *rise* not as predicates of individuals as one might expect but as predicates of individual concepts. For Montague individual concepts were modelled as functions from possible worlds and times to individuals. To say that *rise* holds of an individual concept does not entail that *rise* holds of the individual that the concepts finds at a given world and time. Our strategy is closely related to Montague's. However, instead of using individual concepts we will use frames. By interpreting *rises* as a predicate of frames of type *AmbTemp* as given in (2) we obtain a solution to this puzzle.

(11)
$$\lambda r{:}\left[\mathrm{x}{:}\mathit{Ind}\right]\left(\begin{bmatrix} \text{e-time} & : & \mathit{TimeInt} \\ \mathrm{c_{tns}} & : & \text{e-time}= \iota \\ \mathrm{c_{run}} & : & \text{rise}(r,\text{e-time}) \end{bmatrix}\right)$$

Note that a crucial difference between (10) and (11) is that the first argument to the predicate 'rise' is the complete frame $r$ rather than the value of the x-field which is used for 'run'. Thus it will not follow that the value of the x-field (i.e. 90 in Montague's example) is rising. While there is a difference in the type of the argument to the predicates (a record as opposed to an individual), the type of the complete verb content is the same: $\left[\mathrm{x}{:}\mathit{Ind}\right]\rightarrow\mathit{RecType}$, that is, a function from records of type $\left[\mathrm{x}{:}\mathit{Ind}\right]$ to record types.

## *4.2 Additions to the grammatical resources

The aim of this section is to add to the resources described in section *3.5 so that we can analyze sentences such as *the temperature rises* and *the price rises*.

### Lexicon

*Intransitive verbs*

The ability to use different types internally but still have the same overall type for the content of the word means that we can incorporate verbs that take frame arguments into the lexicon without having to change the rest of the grammar resources. We add a paradigm **rise**:

**rise**($\left[\text{tns=inf}\right]$)= "rise"
**rise**($\left[\text{tns=pres}\right]$)= "rises"
**rise**($\left[\text{tns=past}\right]$)= "rose"

We now introduce a lexical function $\text{lex}_{\text{V}_\text{i}-\text{fr}}$ to be a function which takes a paradigm $\mathcal{W}$ corresponding to a verb whose predicate takes a frame argument, such as **rise**, a predicate $p$ with arity $\langle\left[\text{x:}\mathit{Ind}\right], \mathit{TimeInt}\rangle$ and morphological record $m$ of type $\left[\text{tns:}\mathit{Tns}\right]$ such that

1. if $m$ is $\left[\text{tns=inf}\right]$, $\text{lex}_{\text{V}_\text{i}\text{-fr}}(\mathcal{W}, p, m)$ is

$$\mathit{Sign} \wedge \begin{bmatrix} \text{s-event:}\left[\text{phon:}\mathcal{W}(m)\right] \\ \text{synsem:}\begin{bmatrix} \text{cat=v}_\text{i}\text{:}\mathit{Cat} \\ \text{fin=0:}\mathit{Bool} \\ \text{cnt=}\lambda r\text{:}\left[\text{x:}\mathit{Ind}\right]\ (\begin{bmatrix} \text{e-time:}\mathit{TimeInt} \\ \text{c}_{\mathcal{W}(m)}\text{:}\langle\lambda v\text{:}\mathit{TimeInt}(p(r,v)),\ \langle\text{e-time}\rangle\rangle \end{bmatrix}\text{):}\mathit{Ppty} \end{bmatrix} \end{bmatrix}$$

2. if $m$ is $\left[\text{tns=pres}\right]$, $\text{lex}_{\text{V}_\text{i}\text{-fr}}(\mathcal{W}, p, m)$ is

$$\mathit{Sign} \wedge \begin{bmatrix} \text{s-event:}\begin{bmatrix} \text{phon:}\mathcal{W}(m) \\ \text{s-time:}\mathit{TimeInt} \end{bmatrix} \\ \text{synsem:}\begin{bmatrix} \text{cat=v}_\text{i}\text{:}\mathit{Cat} \\ \text{fin=1:}\mathit{Bool} \\ \text{cnt=}\langle\lambda v_1\text{:}\mathit{Time}\{\lambda r\text{:}\left[\text{x:}\mathit{Ind}\right] \\ \qquad (\begin{bmatrix} \text{e-time:}\mathit{TimeInt} \\ \text{c}_{\mathcal{W}(m)}\text{:}\langle\lambda v_2\text{:}\mathit{TimeInt}(p(r,v_2)),\ \langle\text{e-time}\rangle\rangle \end{bmatrix} \wedge \mathbf{Pres}(v_1))\}, \\ \qquad\qquad \langle\text{s-event.s-time}\rangle\rangle\text{:}\mathit{Ppty} \end{bmatrix} \end{bmatrix}$$

3. if $m$ is $\left[\text{tns=past}\right]$, $\text{lex}_{\text{V}_\text{i}\text{-fr}}(\mathcal{W}, p, m)$ is

$$\textit{Sign} \wedge \begin{bmatrix} \text{s-event:} \begin{bmatrix} \text{phon:}\mathcal{W}(m) \\ \text{s-time:}\mathit{TimeInt} \end{bmatrix} \\ \text{synsem:} \begin{bmatrix} \text{cat=v}_\text{i}\text{:}\mathit{Cat} \\ \text{fin=1:}\mathit{Bool} \\ \text{cnt=}\langle \lambda v_1\text{:}\mathit{Time}\{\lambda r\text{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix} \\ \qquad (\begin{bmatrix} \text{e-time:}\mathit{TimeInt} \\ \text{c}_{\mathcal{W}(m)}\text{:}\langle \lambda v_2\text{:}\mathit{TimeInt}(p(r,v_2)),\ \langle\text{e-time}\rangle\rangle \end{bmatrix} \wedge \mathbf{Past}(v_1))\}, \\ \qquad\qquad \langle\text{s-event.s-time}\rangle\rangle\text{:}\mathit{Ppty} \end{bmatrix} \end{bmatrix}$$

An example of a set of lexical intransitive verb types which could now be generated with these resources given appropriate predicates 'run', 'walk' and 'rise' is

$$\bigcup_{\alpha\in\{\text{inf,pres,past}\}} \{\begin{array}{l} \text{lex}_{\text{V}_\text{i}}(\textsf{run},\text{run},\begin{bmatrix}\text{tns=}\alpha\end{bmatrix}), \\ \text{lex}_{\text{V}_\text{i}}(\textsf{walk},\text{walk},\begin{bmatrix}\text{tns=}\alpha\end{bmatrix}), \\ \text{lex}_{\text{V}_\text{i}\text{-fr}}(\textsf{rise},\text{rise},\begin{bmatrix}\text{tns=}\alpha\end{bmatrix})\} \end{array}$$

*Common nouns*

In our treatment of common nouns we will make the same distinction that we made for intransitive verbs between predicates that take individual arguments and those that take frame arguments.

We define a function $\text{lex}_\text{N}$ which maps phonological types corresponding to common nouns like *dog* and predicates with arity $\langle \mathit{Ind}, \mathit{TimeInt}\rangle$, such that if $W$ is a phonological type like "dog" and $p$ is a predicate with arity $\langle \mathit{Ind}, \mathit{TimeInt}\rangle$, $\text{lex}_\text{N}(W,p)$ is

$$\textit{Sign} \wedge \begin{bmatrix} \text{s-event:}\begin{bmatrix}\text{phon:}W\end{bmatrix} \\ \text{synsem:} \begin{bmatrix} \text{cat=n:}\mathit{Cat} \\ \text{cnt=}\lambda r\text{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\ (\begin{bmatrix} \text{e-time:}\mathit{TimeInt} \\ \text{c}_W\text{:}\langle \lambda v\text{:}\mathit{TimeInt}(p(r\text{.x},v),\ \langle\text{e-time}\rangle\rangle \end{bmatrix}))\text{:}\mathit{Ppty} \end{bmatrix} \end{bmatrix}$$

We define a function $\text{lex}_\text{N-fr}$ which maps phonological types corresponding to common nouns like *temperature* and *price* and predicates with arity $\langle\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}, \mathit{TimeInt}\rangle$, such that if $W$ is a phonological type like "temperature" or "price" and $p$ is a predicate with arity $\langle\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}, \mathit{TimeInt}\rangle$, $\text{lex}_\text{N-fr}(W,p)$ is

$$\textit{Sign} \wedge \begin{bmatrix} \text{s-event:}\begin{bmatrix}\text{phon:}W\end{bmatrix} \\ \text{synsem:} \begin{bmatrix} \text{cat=n:}\mathit{Cat} \\ \text{cnt=}\lambda r\text{:}\begin{bmatrix}\text{x:}\mathit{Ind}\end{bmatrix}\ (\begin{bmatrix} \text{e-time:}\mathit{TimeInt} \\ \text{c}_W\text{:}\langle \lambda v\text{:}\mathit{TimeInt}(p(r,v),\ \langle\text{e-time}\rangle\rangle \end{bmatrix}))\text{:}\mathit{Ppty} \end{bmatrix} \end{bmatrix}$$

An example of a set of lexical common noun types which could be generated given appropriate predicates 'dog', 'temperature' and 'price' is

$\{\text{lex}_{\text{N}}(\text{“dog”},\text{dog}),$
$\text{lex}_{\text{N-fr}}(\text{“temperature”},\text{temperature}),$
$\text{lex}_{\text{N-fr}}(\text{“price”},\text{price})\}$

*Determiners*

We define a function $\text{lex}_{\text{Det-ex}}$ for the indefinite article which maps a phonological type like *a* to a sign type such that if $W$ is a phonological type $\text{lex}_{\text{Det-ex}}(W)$ is

$$\begin{array}{l}\mathit{Sign} \wedge_{\cdot} \\ \left[\begin{array}{ll}\text{s-event:} & \left[\text{phon:}W\right] \\ \text{synsem:} & \left[\begin{array}{l}\text{cat=Det:}\mathit{Cat} \\ \text{cnt=}\ \lambda v_1{:}\mathit{Ppty} \\ \qquad (\lambda v_2{:}\mathit{Ppty} \\ \qquad\quad (\left[\begin{array}{l}\text{par:}\left[\text{x:}\mathit{Ind}\right] \\ \text{restr:}\langle\lambda v{:}\left[\text{x:}\mathit{Ind}\right](v_1(v)),\ \langle\text{par}\rangle\rangle \\ \text{scope:}\langle\lambda v{:}\left[\text{x:}\mathit{Ind}\right](v_2(v)),\ \langle\text{par}\rangle\rangle\end{array}\right])\ ){:}\mathit{Ppty}\rightarrow\mathit{Quant}\end{array}\right]\end{array}\right]\end{array}$$

We define a function $\text{lex}_{\text{Det-uni}}$ for the universal determiner *every* which maps a phonological type such as “every” to a sign type such that if $W$ is a phonological type then $\text{lex}_{\text{Det-uni}}(W)$ is

$$\begin{array}{l}\mathit{Sign} \wedge_{\cdot} \\ \left[\begin{array}{ll}\text{s-event:} & \left[\text{phon:}W\right] \\ \text{synsem:} & \left[\begin{array}{l}\text{cat=Det:}\mathit{Cat} \\ \text{cnt=}\lambda v_1{:}\mathit{Ppty} \\ \qquad (\lambda v_2{:}\mathit{Ppty} \\ \qquad\quad (\left[\begin{array}{l}\text{f:}(r{:}\left[\begin{array}{l}\text{par:}\left[\text{x:}\mathit{Ind}\right] \\ \text{restr:}\langle\lambda v{:}\left[\text{x:}\mathit{Ind}\right](v_1(v)),\ \langle\text{par}\rangle\rangle\end{array}\right]) \\ \qquad\quad \rightarrow v_2(r.\text{par})\end{array}\right])\ ){:}\mathit{Ppty}\rightarrow\mathit{Quant}\end{array}\right]\end{array}\right]\end{array}$$

We define a function $\text{lex}_{\text{Det-def}}$ which maps phonological types to a sign type for the definite article *the* such that if $W$ is an appropriate phonological type then $\text{lex}_{\text{Det-def}}(W)$ is

$$\begin{array}{l}\mathit{Sign} \wedge_{\cdot} \\ \left[\begin{array}{ll}\text{s-event:} & \left[\text{phon:}W\right] \\ \text{synsem:} & \left[\begin{array}{l}\text{cat=Det:}\mathit{Cat} \\ \text{cnt=}\lambda v_1{:}\mathit{Ppty} \\ \qquad (\lambda v_2{:}\mathit{Ppty} \\ \qquad\quad (\left[\begin{array}{l}\text{par:}\left[\text{x:}\mathit{Ind}\right] \\ \text{restr:}\langle\lambda v{:}\left[\text{x:}\mathit{Ind}\right](v_1(v)\wedge_{\cdot} \\ \qquad\quad \left[\begin{array}{l}\text{par:}\left[\text{x:}\mathit{Ind}\right] \\ \text{f:}(r{:}\left[\begin{array}{l}\text{restr:}\langle\lambda v{:}\left[\text{x:}\mathit{Ind}\right](v_1(v)), \\ \qquad\qquad \langle\text{par}\rangle\rangle\end{array}\right]) \\ \qquad\quad \rightarrow\ \left[\text{scope:}v{=}r.\text{par}\right]\end{array}\right] \\ \qquad\quad \langle\text{par}\rangle\rangle) \\ \text{scope:}\langle\lambda v{:}\left[\text{x:}\mathit{Ind}\right](v_2(v)),\ \langle\text{par}\rangle\rangle\end{array}\right],\ ){:}\mathit{Ppty}\rightarrow\mathit{Quant}\end{array}\right]\end{array}\right]\end{array}$$

An example of a set of lexical determiner types that could be generated with these resources is

$\{\text{lex}_{\text{Det-ex}}(\text{“a”}),$
$\text{lex}_{\text{Det-uni}}(\text{“every”}),$
$\text{lex}_{\text{Det-def}}(\text{“the”})\}$

This is a classical treatment of quantification which uses existential quantification similar to that used in classical DRT [Kamp and Reyle, 1993] where existential quantification introduces a discourse referent corresponding to our par(ameter)-field. The indefinite article introduces three fields: the parameter field for the witness of the quantifier, a restriction field corresponding to the common noun following the determiner and a scope field representing the scope of the quantifier, the verb phrase if the noun phrase built with the determiner is in subject position. The result of applying the content of the determiner to two properties will be a type which requires there to be an individual which meets the conditions provided by both the restriction and the scope.

Also similar to classical DRT is the use of dependent functions (as defined on p. 13) for universal quantification. The type resulting from the application of the determiner content to two properties requires that there be a function from individuals meeting the restriction type to a proof that these individuals also meet the restriction. The use of dependent function types for universal quantification is a classical strategy in the application of type theory to natural language semantics. [Sundholm, 1986; Ranta, 1994, are examples of discussion of this].

The definite article content combines the content of both the existential and the universal quantifier contents in the kind of Russellian treatment of definite descriptions that Montague proposed. Applying this content to two properties will return a type which requires that there is some individual which has the first property and that anything which has this property is identical with this individual (i.e. there is exactly one individual with this property) and furthermore the individual also has the second property.

This treatment of quantification (like Montague's) does not use generalized quantifier relations even though the determiner contents are functions which apply to two properties. [Cooper, 2010b, contains discussion of this issue].

*Syntactic and semantic composition*

We need one additional rule to combine determiners and nouns into noun phrases. This rule is similar to the rule combining noun phrases and verb phrases into sentences except that it uses different categories and content types and lacks the finite head requirement.

**NP → Det N**

$$\begin{array}{l}\mathsf{binary\_sign} \wedge \mathsf{phon\_concat} \wedge \mathsf{binary\_cat}(\text{det})(\text{n})(\text{np}) \\ \qquad\qquad \wedge \mathsf{cnt\_forw\_app}(Ppty)(Quant)\end{array}$$

## 5 LEXICAL SEMANTICS, FRAMES AND FERNANDO EVENT STRINGS

### *5.1 Frames that rise*

In the previous section we proposed to solve the Partee puzzle by allowing predicates such as 'rise' to take frames as arguments. But now the question arises: what can it mean for a frame to rise?

In an important series of papers including [Fernando, 2004; Fernando, 2006; Fernando, 2008; Fernando, 2009], Fernando introduces a finite state approach to event analysis where events can be seen as strings of punctual observations corresponding to the kind of sampling we are familiar with from audio technology and digitization processing in speech recognition. When talking about the intuition behind this analysis Fernando sometimes refers to strings of frames in a movie (e.g. in [Fernando, 2008]). But in many cases what he is calling a movie frame can also be seen as a frame in the sense of this paper as well. Thus an event of a rise in temperature could be seen as a concatenation of two temperature frames, that is, an object of type $AmbTemp^\frown AmbTemp$. (12) shows a type of event for a rise in temperature using the temperature frame *AmbTemp* in (2).

(12)
$$\left[\begin{array}{l}\text{e-time}:\mathit{TimeInt}\\ \text{start}:\left[\begin{array}{l}\text{x}:\mathit{Ind}\\ \text{e-time=e-time.start}:\mathit{Time}\\ \text{e-location}:\mathit{Loc}\\ \text{c}_{\text{temp\_at\_in}}:\text{temp\_at\_in(start.e-time, start.e-location, start.x)}\end{array}\right]\\ \text{end}:\left[\begin{array}{l}\text{x}:\mathit{Ind}\\ \text{e-time=e-time.end}:\mathit{Time}\\ \text{e-location=start.e-location}:\mathit{Loc}\\ \text{c}_{\text{temp\_at\_in}}:\text{temp\_at\_in(end.e-time, end.e-location, end.x)}\end{array}\right]\\ \text{event=start}^\frown\text{end}:\mathit{AmbTemp}^\frown\mathit{AmbTemp}\\ \text{c}_{\text{incr}}:\text{start.x<end.x}\end{array}\right]$$

We will call this type *TempRise*. Now we can say something more precise about the content of *rise* expressed in (11). Recall that this introduces a predicate 'rise' which takes a frame and a time interval as arguments. Combinining 'rise' with two arguments creates a type of event in which the frame "rises" during the time interval. We suggest that a candidate for such an event is an object of type *TempRise*. We will express this by

> if $r$:*AmbTemp* and $i$:*TimeInt* then $e$:rise($r$,$i$) iff $e$:*TempRise*, $e$.start=$r$ and $e$.e-time=$i$.

This is at most a very partial account of what objects belong to the types which are constructed from the predicate 'rise'. It is limited to ambient temperature frames. It does not tell us what it would mean for any other kind of frame to rise. This incompleteness is, we believe, an important part of a cognitive theory based on type theory.

Our idea is to exploit an important aspect of the formal development of type theory in a cognitive theory of concept acquisition. We want to say that concepts are modelled by types in type theory. In the formal treatment of type theory when we introduce a new type or a predicate which combines with arguments to form a type there are always two things that have to be done. Firstly the type or predicate itself has to be introduced and we have to say what the type is or how types can be constructed using the predicate. Secondly we have to say what objects belong to the type(s) we have introduced. In the cognitive theory we want to say that we are often (perhaps standardly) in the situation where we have a type or predicate in our cognitive resources (that is, we have performed the counterpart of the first part of type introduction) but we have only a partial idea of what it means to be of the type(s) introduced (that is, we have not been able to complete the second part of the introduction). In fact, we will argue below that at least in the case of concepts corresponding to word meaning we can never be sure that we have a complete account of what it means to belong to the corresponding types.

Thus suppose that we have an agent who has just observed an utterance of the sentence *the temperature rises* and that the utterance of the word *rises* was the first time that the agent had heard this word. From various pieces of evidence the agent may be able to figure out that this is an intransitive verb, for example from the present tense morphology and its position in the sentence. This will provide the agent with enough information to construct a predicate 'rise' given the linguistic resources at the agent's disposal and will enable the agent to conclude that the content of the verb is (11), possibly with a question mark over whether the first argument to the predicate is the whole frame or the x-component of the frame. It is perhaps safer to assume first the more general case where the argument is the whole frame unless there is evidence for the more specific case.

If the agent is at a loss for what was being communicated this might be as far as she can get. That is, she will know that there is a predicate signified by *rise* but she will not have any idea of what it means for a event to fall under a type which is constructed with this predicate. However, there is very often other evidence besides the speech event which will give clues as to what it might mean for a temperature to rise. The agent, herself, may have noticed that it is getting hotter. The speaker of the utterance may indicate by wiping their brow or fanning their face as they speak that this is what is meant by *rise*. (Just think of the kind of gesticulations that often accompany utterances made to non-native speakers of a language with only basic competence.) Thus the agent may come to the account we have presented of what rising involves for ambient temperature situations. A question is: will this account generalize to other kinds of situations?

## *5.2 Word meaning in flux*

For all (12) is based on a very much simplified version of FrameNet's `Ambient_temperature`, it represents a quite detailed account of the lexical meaning of *rise* in respect of ambient temperature — detailed enough, in fact, to make

it inappropriate for *rise* with other kinds of subject arguments. Consider price. The type of a price rising event could be represented by (13).

(13)
$$\left[\begin{array}{l}
\text{e-time}:\mathit{TimeInt} \\
\text{start}:\left[\begin{array}{l}
\text{x}:\mathit{Ind} \\
\text{e-time=e-time.start}:\mathit{Time} \\
\text{e-location}:\mathit{Loc} \\
\text{commodity}:\mathit{Ind} \\
\text{c}_{\text{price_of_at_in}}:\text{price_of_at_in(start.commodity,} \\
\qquad\qquad \text{start.e-time, start.e-location, start.x)}
\end{array}\right] \\
\text{end}:\left[\begin{array}{l}
\text{x}:\mathit{Ind} \\
\text{e-time=e-time.end}:\mathit{Time} \\
\text{e-location=start.e-location}:\mathit{Loc} \\
\text{commodity=start.commodity}:\mathit{Ind} \\
\text{c}_{\text{price_of_at_in}}:\text{price_of_at_in(end.commodity,} \\
\qquad\qquad \text{end.e-time, end.e-location, end.x)}
\end{array}\right] \\
\text{event=start}^\frown\text{end}:\mathit{Price}^\frown\mathit{Price} \\
\text{c}_{\text{incr}}:\text{start.x<end.x}
\end{array}\right]$$

(13) is similar to (12) but crucially different. A price rising event is, not surprisingly, a string of price frames rather than ambient temperature frames. The type of price frames (*Price*) is given in (14).

(14)
$$\left[\begin{array}{lcl}
\text{x} & : & \mathit{Ind} \\
\text{e-time} & : & \mathit{Time} \\
\text{e-location} & : & \mathit{Loc} \\
\text{commodity} & : & \mathit{Ind} \\
\text{c}_{\text{price_of_at_in}} & : & \text{price_of_at_in(commodity, e-time, e-location, x)}
\end{array}\right]$$

If you look up the noun *price* in FrameNet[7] you find that it belongs to the frame `Commerce_scenario` which includes frame elements for goods (corresponding to our 'commodity') and money (corresponding to our 'x'-field). If you compare the FrameNet frames `Ambient_temperature` and `Commerce_scenario`, they may not initially appear to have very much in common. However, extracting out just those frame elements or roles that are relevant for the analysis of the lexical meaning of *rise* shows a degree of correspondence. They are, nevertheless, not the same. Apart from the obvious difference that the predicate in the constraint field that relates the various roles involves temperature in the one and price in the other, price crucially involves the role for commodity since this has to be held constant across the start and end frames. We cannot claim that a price is rising if we check the price of tomatoes in the start frame and the price of oranges in the end frame.

The fact that we need a different meaning for *rise* depending on whether the subject is a temperature or a price corresponds to a situation which is familiar to us from work on the Generative Lexicon [Pustejovsky, 1995; Pustejovsky, 2006]

[7] accessed 8th April, 2010

where the arguments to words representing functions influence the precise meaning of those words. For example, *fast* means something different in *fast car* and *fast road*, although, of course, the two meanings are related. There are two important questions that arise when we study this kind of data:

- is it possible to extract a single general meaning of words which covers all the particular meanings of the word in context?
- is it possible to determine once and for all the set of particular contextually determined meanings?

Our suspicion is that the answer to both these questions is "no". How exactly should we fill out our analysis in order to get the correct meaning of *rise* for prices? Is it sufficient to check the price at just two points of time? If not, how do we determine how many points need to be checked? Should we place restrictions on the time-span between points which are checked and if so, how can we go about determining the kind of time-span involved? Do we need to check that the rising is monotonic, that is, that there is no point in the period we are checking that the price is lower than it was at an earlier time in the period? And then there is the matter of how space is involved in the meaning. If I say *The price of tomatoes is rising* do I mean the price of tomatoes in a particular shop, a particular city, region or in general wherever tomatoes are sold? This seems like a pragmatic dependence on context. But suppose we have determined a region we are interested in. Does the price of tomatoes have to be rising in every shop selling tomatoes in that region or for every kind of tomato? If not, what percentage of the tomatoes in the region need to be going up in price in order for the sentence to be true? This is perhaps a matter having to do with vagueness or generic interpretations. Then there are more technical questions like: is the price rising if it is keeping pace with some recognized index of inflation? Well, it depends what you mean by *rise*. Can the price of tomatoes be said to be rising if it stays the same during a period of deflation?

It seems unlikely that we could tie down the answer to all of these questions once and for all and characterize *the* meaning of *rise*. The techniques we have for dealing with context dependence and vagueness may account for some of the apparent variability, but in the end surely we have to bite the bullet and start building theories that come to grips with the fact that we adjust the meanings of words to fit the purposes at hand.

It seems that we are able to create new meanings for words based on old meanings to suit the situation that we are currently trying to describe and that there is no obvious requirement that all these meanings be consistent with each other, making it difficult to extract a single general meaning. Here we are following the kind of theory proposed by Larsson and Cooper [Larsson and Cooper, 2009; Cooper and Larsson, 2009]. According to such a theory the traditional meaning question "What is the meaning of expression $E$?" should be replaced by the following two questions relating to the way in which agents coordinate meaning as

they interact with each other in dialogue or, more indirectly, through the writing and reading of text:

**the coordination question** Given resources $R$, how can agent $A$ construct a meaning for a particular utterance $U$ of expression $E$?

**the resource update question** What effect will this have on $A$'s resources $R$?

Let us look at a few examples of uses of the verb *rise* which suggest that this is the kind of theory we should be looking at. Consider first that a fairly standard interpretation of *rise* concerns a change in location. (15) is part of the description of a video game.[8]

(15) As they get to deck, they see the Inquisitor, calling out to a Titan in the seas. **The giant Titan rises through the waves**, shrieking at the Inquisitor.

The type of the rising event described here could be something like (16).

(16)
$$\begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{start}: \begin{bmatrix} \text{x}: \mathit{Ind} \\ \text{e-time}=\text{e-time.start}: \mathit{Time} \\ \text{e-location}: \mathit{Loc} \\ \text{c}_{\text{at}}:\text{at(start.x,start.e-location,start.e-time)} \end{bmatrix} \\ \text{end}: \begin{bmatrix} \text{x}=\text{start.x}: \mathit{Ind} \\ \text{e-time}=\text{e-time.end}: \mathit{Time} \\ \text{e-location}: \mathit{Loc} \\ \text{c}_{\text{at}}:\text{at(end.x,end.e-location,end.e-time)} \end{bmatrix} \\ \text{event}=\text{start}^\frown\text{end}:\mathit{Position}^\frown\mathit{Position} \\ \text{c}_{\text{incr}}:\text{height(start.e-location)}<\text{height(end.e-location)} \end{bmatrix}$$

This relies on a frame type *Position* given in (17).

(17)
$$\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{e-time} & : & \mathit{Time} \\ \text{e-location} & : & \mathit{Loc} \\ \text{c}_{\text{at}} & : & \text{at(x,e-location,e-time)} \end{bmatrix}$$

(17) is perhaps most closely related to FrameNet's `Locative_relation`. (16) is structurally different from the examples we have seen previously. Here the content of the 'x'-field, the focus of the frame, which in the case of the verb *rise* will correspond to the subject of the sentence, is held constant in the string of frames in the event whereas in the case of rising temperatures and prices it was the focus that changed value. Here it is the height of the location which increases whereas

[8] `http://en.wikipedia.org/wiki/Risen_(video_game)`, accessed 4th February, 2010

in the previous examples it was important to hold the location constant.[9] This makes it difficult to see how we could give a single type which is general enough to include both varieties and still be specific enough to characterize "the meaning of *rise*". It appears more intuitive and informative to show how the variants relate to each other in the way that we have done.

The second question we had concerned whether there is a fixed set of possible meanings available to speakers of a language or whether speakers create appropriate meanings on the fly based on their previous experience. Consider the examples in (18).

(18) a. Mastercard rises

b. China rises

While speakers of English can get an idea of the content of the examples in (18) when stripped from their context, they can only guess at what the exact content might be. It *feels* like a pretty creative process. Seeing the examples in context as in (19) reveals a lot.[10]

(19) a. Visa Up on Q1 Beat, Forecast; **Mastercard Rises** in Sympathy

By Tiernan Ray

Shares of Visa (V) and Mastercard (MA) are both climbing in the aftermarket, reversing declines during the regular session, after Visa this afternoon reported fiscal Q1 sales and profit ahead of estimates and forecast 2010 sales growth ahead of estimates, raising enthusiasm for its cousin, Mastercard.

b. The rise of China will undoubtedly be one of the great dramas of the twenty-first century. China's extraordinary economic growth and active diplomacy are already transforming East Asia, and future decades will see even greater increases in Chinese power and influence. But exactly how this drama will play out is an open question. Will China overthrow the existing order or become a part of it? And what, if anything, can the United States do to maintain its position as **China rises**?

It seems like the precise nature of the frames relevant for the interpretation of *rises* in these examples is being extracted from the surrounding text by a technique related to automated techniques of relation extraction in natural language

[9] We have used 'height(start/end.e-location)' in (16) to represent the height of the location since we have chosen to treat *Loc*, the type of spatial location, as a basic type. However, in a more detailed treatment *Loc* should itself be treated as a frame type with fields for three coordinates one of them being height, so we would be able to refer to the height of a location $l$ as $l$.height.

[10] `http://blogs.barrons.com/stockstowatchtoday/2010/02/03/visa-up-on-q1-beat-forecast-mastercard-moves-in-sympathy/?mod=rss_BOLBlog`, accessed 4th February, 2010; `http://www.foreignaffairs.com/articles/63042/g-john-ikenberry/the-rise-of-china-and-the-future-of-the-west`, accessed 4th February, 2010.

processing. We know from (19a) that *Mastercard rises* means that the share price of Mastercard has been going up. We have, as far as I can see, no clear way of determining whether this involves an adjustment to the meaning of *rise* or of *Mastercard*. It seems that no harm would arise from both strategies being available to an agent. (19b) is interesting in that the text preceding *China rises* prepares the ground so that by the time we arrive at it we have no trouble figuring out that what is meant here by *rise* has to do with economic growth, active diplomacy, power and influence.

Consider (20) out of context.

(20) dog hairs rise

Unless you know this example from the British National Corpus it is unlikely that you would get at the appropriate meaning for *rise* which becomes obvious when we look at some of the context as in (21).

(21)

| | |
|---|---|
| Cherrilyn: | Yeah I mean ⟨pause⟩ dog hairs rise anyway so |
| Fiona: | What do you mean, rise? |
| Cherrilyn: | The hair ⟨pause⟩ it rises upstairs. |

BNC file KBL, sentences 4201–4203

(21) is an example of a clarification request (as discussed recently in [Ginzburg, forthcoming], and much previous literature cited there). Given that the meaning of *rise* does not appear to be fixed, we might expect that a lot of such clarification requests would occur. This, however, does not appear to be the case. Out of 205 occurrences of *rise* as a verb (in any of its inflectional forms[11]) in the dialogue subcorpus of BNC there is one occurrence of a clarification, namely (21). It seems then that there is no evidence that *rise* is particularly hard to understand, which certainly seems to accord with intuition. It does seem, however, that human speakers are particularly adept at adjusting meaning to suit the needs of the current situation.

## 6 TOWARDS A THEORY OF SEMANTIC COORDINATION

It seems that we are constantly in the process of creating and modifying language as we speak. This phenomenon has been studied and theorized about for at least 25 years by psychologists of language, for example [Clark and Wilkes-Gibbs, 1986; Garrod and Anderson, 1987; Brennan and Clark, 1996; Healey, 1997; Pickering and Garrod, 2004]. However, in semantics (both formal and empirical) there is a

[11] We used the tool SCoRE [Purver, 2001] using the regular expression `<V+>r(i|o)s(e((n|s))?|ing)` to extract our examples, removing erroneously tagged examples by hand.

tradition of abstracting away from this fact for rather obvious reasons. In formal semantics there is an obvious advantage of assuming that natural languages have a fixed semantics like formal languages in order to get started building a theory of compositional semantics. In empirically based semantics like frame semantics based on corpus data the point is to make statistically based generalizations over whatever varieties of language might be contained within the corpus. However, recently a number of linguists have become interested in trying to account for the way in which language gets created or adjusted through dialogue interaction. For some examples see [Cooper and Kempson, 2008].

One of the important facts to emerge from the psychological research is that dialogue participants tend to coordinate (or align) their language. For example, if you use a particular word for a particular concept, I will tend to use that word for that concept unless there is good reason for me do otherwise (such as I believe that your way of saying it is incorrect or I wish to mark that I speak a different dialect from you). If I use a different word there will be a tendency for my interlocutor to assume that I am referring to a different concept all else being equal. [Clark, 1993] refers to this as the *principle of contrast*. Similarly, if you use a word with what is for me an innovative meaning, I need to find a way of constructing that meaning so that either we can continue the dialogue using that meaning or we can negotiate what the word should mean. That is, we need to engage in semantic coordination [Larsson, 2007a; Larsson, 2007b]. Suppose you use the word *rise* with a meaning that is new for me. How should I go about figuring out what that meaning should be? One thing that seems clear is that I do not start from scratch, considering the space of all possible intransitive verb meanings. Rather I start from meanings I have previously associated with utterances of *rise* and try to modify them to fit the current case. The structured approach to meaning presented by TTR becomes very important here. Suppose that I have a meaning for *rise* of the classical Montague semantics kind, that is, a function from possible worlds and times to characteristic functions of sets of objects. The kinds of modifications that are natural to such an object could, for example, involve adding or subtracting objects which are associated with a particular world and time. Such modifications are not particularly useful or intuitive in helping us to figure out the answers to the kinds of questions about the meaning of *rise*. In contrast TTR provides us with components that can be modified, parameters which can be used to characterize variation in meaning and serve as a basis for a similarity metric. Components can be modified in order to create new meanings from old.

Our idea is that this view of semantics should be embedded in the kind of view of agents that coordinate linguistic resources which is presented in [Cooper and Larsson, 2009]. We will review the ideas about agents presented there which are in turn based on Larsson's earlier work.

We conceive the theory as being within the gameboard approach to dialogue developed by Ginzburg [Ginzburg, 1994; Ginzburg, forthcoming, and much other literature in between] and the computationally oriented approach based on Ginzburg's work which has come to be known as the information state update

approach [Larsson and Traum, 2001, and much other literature developing from these ideas]. Here dialogue participants have information states associated with the dialogue which are updated by dialogue moves which they perceive as having been carried out by their interlocutors or themselves. The kind of update which this literature has normally been concerned with have to do with both informational content and metalinguistic information about what was said. Informational content includes, for example, propositions which have been committed to in the dialogue and questions under discussion. Metalinguistic information includes information about phonology and syntax as well as what content is to be associated with various parts of the utterance. This provides us with a basis for dealing with part of a theory of semantic coordination. In addition we need to be able to talk about updates to linguistic resources available to the agent (grammar, lexicon, semantic interpretation rules etc. in the sense discussed in [Cooper and Ranta, 2008]) which can take place during the course of a dialogue. The view presented in [Cooper and Larsson, 2009] is that agents have generic resources which they modify to construct local resources for sublanguages for use in specific situations. Thus an agent $A$ may associate a linguistic expression $e$ with a particular concept (or collection of concepts if $e$ is ambiguous) $[\mathrm{e}]^A$ in its generic resource. In this paper, we will think of $[\mathrm{e}]^A$ not as a collection of concepts but as one of the sign types that we introduced above. In a particular domain $\alpha$ $e$ may be associated with a modified version of $[\mathrm{e}]^A$, $[\mathrm{e}]^A_\alpha$ [Larsson, 2007a].

The motor for generating new local resources in an agent lies in coordinating resources with another agent in a particular communicative situation $s$. In a communicative situation $s$, an agent $A$ may be confronted with an *innovative* utterance $e$, that is, an speech event which contains uses of linguistic expressions not already present in $A$'s resources or linguistic expressions from $A$'s resources which in $s$ are associated with an interpretation distinct from that provided by $A$'s resources. In the theory we have presented above either of these cases will involve the construction of a new sign type which is specific to $s$, $[\mathrm{e}]^A_s$, and which may be anchored to the specific objects under discussion in $s$ (using the technique of manifest fields).

Whereas in a standard view of formal grammar there will be one sign (or in our terms sign type) corresponding to a particular interpretation of an expression, we want to see $e$ as related to a whole hierarchy of sign types: $[\mathrm{e}]^A_s$ for communicative situations $s$, $[\mathrm{e}]^A_\alpha$ for domains $\alpha$ (where we imagine that the domains are collected into a complex hierarchy or more and less general domains) and ultimately a general linguistic resource which is domain independent, $[\mathrm{e}]^A$. We think of the acquisition of a particular sign type as a progression from $[\mathrm{e}]^A_s$ for some particular communicative situation $s$, through potentially a series of increasingly general domains $\alpha$ yielding resources $[\mathrm{e}]^A_\alpha$. In [Cooper and Larsson, 2009] we regarded complete acquisition process as ultimately leading to a domain independent generic resource, $[\mathrm{e}]^A$. However, the more one thinks in these terms the more likely it seems that there is no ultimate domain independent resource at all (except perhaps for "logical" words like determiners) but rather a large collection of resources

associated with domains of varying generality.

There is no guarantee that any sign type will survive even beyond the particular communicative situation in which $A$ first encountered it. For example, the kind of *ad hoc* coinages described in [Garrod and Anderson, 1987] using words like *leg* to describe part of an oddly shaped maze in the maze game probably do not survive beyond the particular dialogue in which they occur. The factors involved in determining how a particular sign-type progresses we see as inherently stochastic with parameters including the degree to which $A$ regards their interlocutor as an expert, how many times the sign type has been exploited in other communicative situations and with different interlocutors, the utility of the interpretation in different communicative situations, and positive or negative feedback obtained when exploiting the sign type in a communicative situation. For example, a particular agent may only allow a sign type to progress when it has been observed in at least $n$ different communicative situations at least $m$ of which were with an interlocutor considered to be an expert, and so on.

On this view the kind of question we need to be addressing in a formal linguistic theory is not so much "What is the meaning of *rises* (with respect to price)?" but rather "How will agent $A$ with access to a resource $[rises]^A_\alpha$ (for domain $\alpha$) exploit this resource in a given communicative situation $s$?". Here we assume that exploiting a resource standard involves modifying it so that it matches the purposes at hand. The tradition that we have inherited from logical semantics has given us the idea of truth conditions and determining whether a given sentence is true under the fixed interpretation provided by the language. Here we are also allowing for the option of modifying the interpretation of a sentence so that it would be true in the current state of affairs. If $A$ says $\phi$ to $B$ in respect of situation $s$ and is being honest and is not mistaken about the nature of $s$, then $A$ must be interpreting $\phi$ in such a way that it correctly describes $s$ and it is part of $B$'s task to figure out what this interpretation might be. This is the task of semantic coordination. The challenge for $B$ is to figure out whether there is a reasonable interpretation of $\phi$ (not too different from an interpretation that could be achieved by the resources already at $B$'s disposal based on previous experience) or whether $A$ is in fact mistaken about the nature of $s$ and is saying something false. It seems that much of the misunderstanding that occurs in dialogue can be related to this delicate balancing act that is required of dialogue participants.

We will try to be a little more concrete about what kind of modifications can be made to resources. For the sake of argument we could say that $[rises]^A_\alpha$ is the type which is produced by the grammar we have defined above. The main opportunities for modification here lie in determining what kind of Fernando event strings constitute a rising. If we are talking about price the question is more specifically what strings constitute a rising in price. Since according to this resource the relevant strings are strings of price frames, modifications here may also have consequences for the type of price frames provided by the resource. For example, an agent might soon notice that location is an important parameter for prices (the price might be rising in Europe but not in China, for example). This would mean that strings

of price frames constituting a rising could now be required to contain the same location. This is what we have represented in our type for price rising events.

Suppose that at a given location new higher quality tomatoes become available on the market in addition to the tomatoes that were already available which are still available at the same price. Has the price of tomatoes risen? In one sense, yes, the average price of tomatoes has risen. In another sense, no, people who want to buy the tomatoes they were buying before can continue to do so for the same price. To get the first of these meanings we need to include information about averages in a more detailed price frame. For the second of these meanings we could introduce a parameter for quality into price frames and require that quality be held constant in rising events. These are additions which we may well not want to make part of any general language resource - they are rather *ad hoc* adjustments for the situation at hand. Suppose now that the cheap tomatoes disappear from the market but the more expensive tomatoes are still available for the same price. Again, if what you mean by *price* is the average price then the price has risen. But actually there are no tomatoes on the market such that they have gone up in price. So you could argue (and people do) that prices have not gone up (using price frames with the quality parameter). However, people who need tomatoes for their pasta sauce and who used to buy the cheap tomatoes will now notice a rise in price greater than the average price rise. Whereas they used to get tomatoes for the cheap price they now have to pay the expensive price. For them, the price of tomatoes has risen. Here we seem to need price frames which accommodate a range of prices for a given commodity, for example a price record that specifies the highest and lowest prices, and a characterization of rising in terms of the lowest price. Again, this is an *ad hoc* modification which will be useful for some dialogues but not for others. Once you have figured it out it might be useful to keep in your collection of resources in case you need it again.

An important part of this discussion is that in order to figure out what is meant in a particular speech event we need to match potential interpretations against what we can observe about the world. We observe that $A$ uses $\phi$ to describe situation $s$ and thereby draw conclusions about what $A$ meant by this particular utterance of $\phi$ as well as gaining information about $s$. Perhaps one of the most straightforward examples of this connection is in language acquisition situations where one agent indicates particular objects for another agent saying things like *This is a...*. The challenge for an agent in this learning situation is not so much to determine whether the utterance is true as trying to construct an appropriate meaning for the utterance to make it true and storing this as a resource for future use. Getting this coupling between language and perception is, for example, one of the first challenges in getting a robot to learn language through interaction (see for example the roadmap presented by the ITalk project, `http://www.italkproject.org/`).

## 7 CONCLUSION

One of the great advances in the study of language during the twentieth century was the application of the theory of formal languages to natural language. Challenges for the study of language in the twenty-first century are to extend the formal approach to the study of

1. interaction in dialogue
2. language coordination, including the creation or modification of linguistic resources during the course of a dialogue
3. the relationship of these processes to other cognitive processes such as perception

During the first decade of the century we have made some significant progress on (1), for example, [Ginzburg, forthcoming]. We have also made a start on (2), for example, [Larsson, 2007a; Larsson, 2007b; Cooper and Larsson, 2009]. TTR plays a significant role in this literature. TTR might also be useful in addressing (3) in that type theory is a theory about type judgements which from a cognitive point of view has to do with how we perceive objects.

It is important that we do not lose what we gained during the twentieth century when we are working with these new challenges and we beleive that by using the tools provided by TTR it is plausible that we can keep and improve on the twentieth century canon.

Type theory is appealing for application to the new challenges because it makes the connection between perception and semantics and, with records, provides us with the kind of structure (like frames) that we seem to need for semantic coordination, giving us handles (in the forms of labelled fields) to items of knowledge in a structure rather than the monolithic functions of classical model theoretic semantics.

## BIBLIOGRAPHY

[Austin, 1962] J. Austin. *How to Do Things with Words*. Oxford University Press, 1962. ed. by J. O. Urmson.

[Barwise and Perry, 1983] Jon Barwise and John Perry. *Situations and Attitudes*. Bradford Books. MIT Press, Cambridge, Mass., 1983.

[Barwise, 1989] Jon Barwise. *The Situation in Logic*. CSLI Publications, Stanford, 1989.

[Betarte and Tasistro, 1998] Gustavo Betarte and Alvaro Tasistro. Extension of Martin-Löf's type theory with record types and subtyping. In Giovanni Sambin and Jan Smith, editors, *Twenty-Five Years of Constructive Type Theory*, number 36 in Oxford Logic Guides. Oxford University Press, Oxford, 1998.

[Betarte, 1998] Gustavo Betarte. *Dependent Record Types and Algebraic Structures in Type Theory*. PhD thesis, Department of Computing Science, University of Gothenburg and Chalmers University of Technology, 1998.

[Brennan and Clark, 1996] Susan E. Brennan and Herbert H. Clark. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 22:482–493, 1996.

[Clark and Wilkes-Gibbs, 1986] Herbert H. Clark and D. Wilkes-Gibbs. Refering as a collaborative process. *Cognition*, 22:1–39, 1986.

[Clark, 1993] Eve V. Clark. *The lexicon in acquisition*. Number 65 in Cambridge Studies in Linguistics. Cambridge University Press, Cambridge, 1993.

[Cooper and Kempson, 2008] Robin Cooper and Ruth Kempson, editors. *Language in Flux: Dialogue Coordination, Language Variation, Change and Evolution*, volume 1 of *Communication, Mind and Language*. College Publications, London, 2008.

[Cooper and Larsson, 2009] Robin Cooper and Staffan Larsson. Compositional and ontological semantics in learning from corrective feedback and explicit definition. In Jens Edlund, Joakim Gustafson, Anna Hjalmarsson, and Gabriel Skantze, editors, *Proceedings of DiaHolmia: 2009 Workshop on the Semantics and Pragmatics of Dialogue*, pages 59–66. Department of Speech, Music and Hearing, KTH, 2009.

[Cooper and Ranta, 2008] Robin Cooper and Aarne Ranta. Natural Languages as Collections of Resources. In Cooper and Kempson [2008], pages 109–120.

[Cooper, 2005a] Robin Cooper. Austinian truth, attitudes and type theory. *Research on Language and Computation*, 3:333–362, 2005.

[Cooper, 2005b] Robin Cooper. Records and record types in semantic theory. *Journal of Logic and Computation*, 15(2):99–112, 2005.

[Cooper, 2008] Robin Cooper. Type theory with records and unification-based grammar. In Fritz Hamm and Stephan Kepser, editors, *Logics for Linguistic Structures*, pages 9–34. Mouton de Gruyter, 2008.

[Cooper, 2010a] Robin Cooper. Frames in formal semantics. In Hrafn Loftsson, Eiríkur Rögnvaldsson, and Sigrún Helgadóttir, editors, *IceTAL 2010*. Springer Verlag, 2010.

[Cooper, 2010b] Robin Cooper. Generalized quantifiers and clarification content. In Paweł Łupkowski and Matthew Purver, editors, *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue*. Polish Society for Cognitive Science, Poznań, 2010.

[Coquand *et al.*, 2004] Thierry Coquand, Randy Pollack, and Makoto Takeyama. A logical framework with dependently typed records. *Fundamenta Informaticae*, XX:1–22, 2004.

[Davidson, 1980] Donald Davidson. *Essays on Actions and Events*. Oxford University Press, 1980. New edition 2001.

[Fernando, 2004] Tim Fernando. A finite-state approach to events in natural language semantics. *Journal of Logic and Computation*, 14(1):79–92, 2004.

[Fernando, 2006] Tim Fernando. Situations as strings. *Electronic Notes in Theoretical Computer Science*, 165:23–36, 2006.

[Fernando, 2008] Tim Fernando. Finite-state descriptions for temporal semantics. In Harry Bunt and Reinhart Muskens, editors, *Computing Meaning, Volume 3*, volume 83 of *Studies in Linguistics and Philosophy*, pages 347–368. Springer, 2008.

[Fernando, 2009] Tim Fernando. Situations in LTL as strings. *Information and Computation*, 207(10):980–999, 2009.

[Fillmore, 1982] Charles J. Fillmore. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul, 1982.

[Fillmore, 1985] Charles J. Fillmore. Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2):222–254, 1985.

[Garrod and Anderson, 1987] Simon C. Garrod and Anthony Anderson. Saying what you mean in dialogue: a study in conceptual and semantic co-ordination. *Cognition*, 27:181–218, 1987.

[Ginzburg, 1994] Jonathan Ginzburg. An update semantics for dialogue. In Harry Bunt, editor, *Proceedings of the 1st International Workshop on Computational Semantics*, Tilburg University, 1994. ITK Tilburg.

[Ginzburg, forthcoming] Jonathan Ginzburg. *The Interactive Stance: Meaning for Conversation*. Oxford University Press, Oxford, forthcoming.

[Healey, 1997] P.G.T. Healey. Expertise or expertese?: The emergence of task-oriented sublanguages. In M.G. Shafto and P. Langley, editors, *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, pages 301–306, 1997.

[Jurafsky and Martin, 2009] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson Education, second edition, 2009.

[Kamp and Reyle, 1993] Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Kluwer, Dordrecht, 1993.

[Larsson and Cooper, 2009] Staffan Larsson and Robin Cooper. Towards a formal view of corrective feedback. In Afra Alishahi, Thierry Poibeau, and Aline Villavicencio, editors, *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 1–9. EACL, 2009.

[Larsson and Traum, 2001] Staffan Larsson and David R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3&4):323–340, 2001.

[Larsson, 2007a] Staffan Larsson. Coordinating on ad-hoc semantic systems in dialogue. In R. Arnstein and L. Vieu, editors, *Proceedings of DECALOG - The 2007 Workshop on the Semantics and Pragmatics of Dialogue*, pages 109–116, 2007.

[Larsson, 2007b] Staffan Larsson. A general framework for semantic plasticity and negotiation. In Harry Bunt and E. C. G. Thijsse, editors, *Proceedings of the 7th International Workshop on Computational Semantics (IWCS-7)*, pages 101–117, 2007.

[Linell, 2009] Per Linell. *Rethinking Language, Mind, and World Dialogically: Interactional and contextual theories of human sense-making*. Advances in Cultural Psychology: Constructing Human Development. Information Age Publishing, Inc., Charlotte, N.C., 2009.

[Martin-Löf, 1984] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Naples, 1984.

[Montague, 1973] Richard Montague. The Proper Treatment of Quantification in Ordinary English. In Jaakko Hintikka, Julius Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 247–270. D. Reidel Publishing Company, Dordrecht, 1973.

[Montague, 1974] Richard Montague. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, 1974. ed. and with an introduction by Richmond H. Thomason.

[Pickering and Garrod, 2004] M.J. Pickering and S. Garrod. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27(02):169–190, 2004.

[Prior, 1957] Arthur N. Prior. *Time and modality*. Oxford University Press, 1957.

[Prior, 1967] Arthur N. Prior. *Past, present and future*. Oxford University Press, 1967.

[Purver, 2001] Matthew Purver. SCoRE: A tool for searching the BNC. Technical Report TR-01-07, Department of Computer Science, King's College London, October 2001.

[Pustejovsky, 1995] James Pustejovsky. *The Generative Lexicon*. MIT Press, Cambridge, Mass., 1995.

[Pustejovsky, 2006] James Pustejovsky. Type theory and lexical decomposition. *Journal of Cognitive Science*, 6:39–76, 2006.

[Ranta, 1994] Aarne Ranta. *Type-Theoretical Grammar*. Clarendon Press, Oxford, 1994.

[Reichenbach, 1947] Hans Reichenbach. *Elements of Symbolic Logic*. University of California Press, 1947.

[Sag *et al.*, 2003] Ivan A. Sag, Thomas Wasow, and Emily M. Bender. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, 2nd edition, 2003.

[Searle, 1969] John R. Searle. *Speech Acts: an Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[Seligman and Moss, 1997] Jerry Seligman and Larry Moss. Situation theory. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*. North Holland and MIT Press, 1997.

[Sundholm, 1986] Göran Sundholm. Proof theory and meaning. In Dov Gabbay and Franz Guenthner, editors, *Handbook of Philosophical Logic, Vol. III*. Reidel, Dordrecht, 1986.

[Tasistro, 1997] Alvaro Tasistro. *Substitution, record types and subtyping in type theory, with applications to the theory of programming*. PhD thesis, Department of Computing Science, University of Gothenburg and Chalmers University of Technology, 1997.

[Trubetzkoy, 1939] N.S. Trubetzkoy. *Grundzüge der Phonologie*. Vandenhoeck and Ruprecht, Göttingen, 1939.

[Turner, 2005] Raymond Turner. Semantics and stratification. *Journal of Logic and Computation*, 15(2):145–158, 2005.

[Wittgenstein, 1922] Ludwig Wittgenstein. *Tractatus Logico-Philosophicus*. Routledge and Kegan Paul, 1922. translated by C.K. Ogden.

[Wittgenstein, 1953] Ludwig Wittgenstein. *Philosophical Investigations*. Blackwell, Oxford, 1953.