Phrase structure rules as dialogue update rules

Robin Cooper
University of Gothenburg
cooper@ling.gu.se

Abstract

We present a formulation of phrase structure rules in TTR (Type Theory with Records (Cooper, 2012)) as dialogue update rules of a similar kind to those discussed by Ginzburg (2012) and Larsson (2002). This grounds syntax in a theory of events. Apart from unifying syntax with a theory of dialogue processing, there are two main advantages to the proposal: (1) it places constraints on a natural non-abstract theory of syntax which represents linguistic events as they occur and (2) it points us to an account of incremental interpretation in terms of the processing of strings of events in a manner similar to that proposed by Poesio and Traum (1997) and Poesio and Rieser (2010).

1 Introduction

A common view of how dialogue analysis fits into linguistic theory is that dialogue comes as a superordinate structure built on top of syntax, semantics and the other conceptual components of linguistic theory where the kinds of tools used in dialogue analysis seem quite different to what is needed for the other components. I want to suggest that we can turn this around: that everything in linguistic analysis can be thought of in terms of the tools we need for dialogue, that is, tools required for the analysis of communication involving the perception and creation of types of linguistic events and reasoning about updates to information states. And I want to suggest that we can pursue this idea without sacrificing the kind of formal rigour we are able to achieve in more traditional approaches to linguistic analysis.

In this paper we show that we can view phrase structure rules in TTR (Type Theory with Records (Cooper, 2012)) as dialogue update rules. In this way I want to suggest that the foundational notions which run through all the components of linguistic theory have to do with the perception and creation of communicative events in the way that has been discussed in formal theories of dialogue such as Ginzburg (2012) and Larsson (2002). The ideas presented in this paper could be regarded as implicit in their work, although they do not make an explicit connection with phrase structure. There are three aspects of our particular approach which we would like to highlight:

1. Grounding syntax in event perception and creation places intuitive restrictions on what a "natural" syntax is and makes abstract theories of syntax with many inaudible constituents appear as a rather different kind of theory.

2. It also points the way to a view of incremental parsing as information state update in a way which is related to proposals by Poesio and Traum (1997) and Poesio and Rieser (2010).

3. The use of TTR enables us to factor the phrase structure rules into various abstract resources which can be combined. The result gives us a view of universal grammar similar to that of Jackendoff (2002) and Cooper and Ranta (2008) where linguistics universals are regarded as a kind of toolbox from which natural languages select.

The approach we are taking also has a lot in common with that taken by Purver et al. (2010) and Eshghi et al. (2012). There the strategy is to incorporate TTR into Dynamic Syntax. Here the strategy is to incorporate ideas from Dynamic Syntax into TTR. Another related approach which needs to be explored in this connection is represented by Demberg et al. (2013).

We will first present a particular view of update in terms of TTR and then we will show how phrase structure rules can be considered in these terms. We will not give a detailed introduction to TTR notation[1], although we will use it liberally for the sake of concreteness. However, we will give an intuitive explanation of each formula which should make the ideas accessible to readers unfamiliar with the details of the notation.

We shall consider some of the resources needed to deal with a very simple toy dialogue as in (1).

(1)	User:	Dudamel is a conductor
	System:	Aha
	User:	Beethoven is a composer
	System:	OK

2 Update functions

We will assume that agents do not have complete information about their information state, that is, they reason in terms of *types* of information state (that is, gameboards). The basic intuition behind our reasoning about information state updates can be expressed as in (2).

(2) If $r_i : T_i$, then $r_{i+1} : T_{i+1}(r_i)$

That is, given that we believe that the current information state is of type T_i, then we can conclude that the next information state is of type T_{i+1} which can depend on the current information state. According to this, we can have a hypothesis about the type of the next information state even though we may not know exactly what the current information state is. Thus the dependency in our types provides us with a means for representing underspecification.

This basic rule of inference corresponds to a function from records to record types, a function of type ($T_i \rightarrow$ *RecType*). Such a function is of the form (3).

(3) $\lambda r\!:\!T_i \,.\, T_{i+1}(r)$

Things are a litte more complicated than this, however, because this only represents the change from one information state to another, whereas in fact this change is triggered by an event (speech or otherwise) which bears an appropriate relation to the current information state represented by r. Thus we are actually interested in functions from the current information state to a function from events to the new information state, as in (4).

(4) $\lambda r\!:\!T_i \,.\, \lambda e\!:\!T_e(r) \,.\, T_{i+1}(r, e)$

This is one of a number of ways of characterizing update in this kind of framework. One might for instance think of the type of the speech event as being part of the current information state. Also instead of using an update function one can use a record type with a 'preconditions'-field and an 'effect'-field. Both Ginzburg (2012) and Larsson (2002) have this kind of approach. Our formulation makes explicit that update functions are dependent types, that is functions from objects (including information states and events) to a *type*, in this case for the updated information state. We will see that this makes clear a natural relationship between update functions and phrase structure rules viewed as functions (similar to a categorial grammar approach).

Let us consider the update function which the user could use in order to update her information state after her own utterance of *Dudamel is a conductor*. The function in Figure 1 is modelled on the kind of integration rules discussed in (Larsson, 2002). This function maps information states (records), r, which have a non-empty agenda to a function that maps events to a type of information state. It thus requires that the current information state (the first argument to the function) have a non-empty agenda. The second argument to the function (represented by u) requires the move associated with the speech-event to be of the first type on the agenda in r, the current information state, and also to be an assertion with *SELF* as the speaker. It also requires that the chart associated with this utterance can be interpreted as a move of that type. The requirements on the arguments to the function represent the preconditions. The type that results from applying the function to its arguments represents the effect of the update. This type requires the agenda to be the result of replacing the first type on the agenda in r with an acknowledgement where the speaker is the audience of the assertion move and the audience of the acknowledgement is *SELF*. The content of the acknowledgement is the same as the content of the assertion. That is, what is being acknowledged is the content of the assertion. It furthermore requires

[1]This can be found in Cooper (2012) and in the updated drafts of a manuscript in progress called *Type theory and language: from perception to linguistic communication* to be found on `https://sites.google.com/site/typetheorywithrecords/drafts`.

$$
\begin{array}{l}
\lambda r{:}\left[\begin{array}{lll}\text{private} & : & \left[\begin{array}{lll}\text{agenda} & : & {}_{ne}[\mathit{MoveType}(\mathit{SELF})]\end{array}\right]\end{array}\right] \\
\quad\lambda u{:}\left[\begin{array}{lll}
\text{move} & : & \text{fst}(r.\text{private.agenda}) \mathbin{\dot\wedge} \left[\text{e:}\left[\begin{array}{l}\text{sp=}\mathit{SELF}{:}\mathit{Ind}\\ \text{au:}\mathit{Ind}\end{array}\right]\right] \mathbin{\dot\wedge} \left[\text{e:}\mathit{Assertion}\right] \\
\text{chart} & : & \mathit{Chart} \\
\text{e} & : & \text{m-interp(chart,move)}
\end{array}\right] . \\
\quad\left[\begin{array}{ll}
\text{private:} & \left[\begin{array}{ll}\text{agenda=} & \begin{array}{l}\left[\begin{array}{l}\text{e:}\mathit{Acknowledgement} \mathbin{\dot\wedge} \left[\begin{array}{l}\text{sp=}u.\text{move.e.au:}\mathit{Ind}\\ \text{au=}\mathit{SELF}{:}\mathit{Ind}\end{array}\right] \\ \text{cnt=}u.\text{move.cnt:}\mathit{RecType} \\ \text{c}_{\text{cnt}}\text{:content(e,cnt)}\end{array}\right] \\ \qquad\qquad\qquad\mid \text{rst}(r.\text{private.agenda})\end{array} :[\mathit{MoveType}(\mathit{SELF})]\end{array}\right] \\
\text{shared:} & \left[\begin{array}{ll}\text{latest-utterance:} & \left[\begin{array}{l}\text{move=}u.\text{move:}\mathit{Move}(\mathit{SELF}) \\ \text{chart=}u.\text{chart:}\mathit{Chart} \\ \text{e=}u.\text{e:m-interp(chart,move)}\end{array}\right]\end{array}\right]
\end{array}\right]
\end{array}
$$

Figure 1

the latest-utterance field to contain the move and chart of the utterance u. The idea is that this function should be used to predict the type of the next information state on the basis of the current information state and the observed event. That is, if we believe the current information state to be of the domain type of the update function and we observe an event of the required type then we reason that the updated information state should be of the type resulting from applying the function to the current information state.

We will now examine how such an update function could be used to reason about an update. Let us suppose that the user considers the current information state to be of type Figure 2.

This represents that the user intends to assert that Dudamel is a conductor represented by the record type $\left[\text{e:conductor(Dudamel)}\right]$. The user also believes that there was no previous utterance and no commitments, i.e. that the planned utterance will be dialogue initial.

Suppose now that the user utters *Dudamel is a conductor* and judges this utterance event u_1 to be an event of type Figure 3.

The user will have more information about the nature of the chart (that is, about what was actually said and how it might be analyzed) than we have represented but we will leave this underspecified.

Clearly in the user's judgement the utterance u_1 fulfils the requirements placed on it by Figure 1 since the move interpretation associated with it is of the type which occurs at the head of the agenda. Note that we are reasoning with this function without actually providing it with an argument since we only have a (hypothesized) type of the current information state, not the actual information state. The crucial judgement is that the type of the current information state is a subtype of the domain type of the function. This is sufficient to allow us to come to a conclusion about the type of the new information state.

According to the update function the next information state must be of the type Figure 4. Note that the speaker in the type on the agenda here is the audience of the original utterance. Thus what is on the agenda is a type of act to be carried out by the interlocutor rather than the *SELF*. This is a way of implementing simple turn-taking in a gameboard approach to dialogue. It also represents the fact that the realization of event types is often a collaborative process. An utterance is not successfully acknowledged if the person who made the original utterance is no longer paying attention, for example.

But we know more about the new information state than what is expressed by the type which results from the update function. Everything we know about the current information state which remains unchanged by the function must be carried over from the current information state. This is related to the frame problem introduced by (McCarthy and Hayes, 1969).[2] We handle this by performing an *asymmetric merge* of the type we have for the current information state with the type resulting from the update function. The asymmetric merge of two types T_1 and T_2 is represented by

[2]For a recent overview of the frame problem see (Shanahan, 2009).

$$\left[\begin{array}{ll} \text{private:} & \left[\text{agenda=[} \left[\begin{array}{l} \text{e:}\mathit{Assertion} \underset{\cdot}{\wedge} \left[\text{sp=}\mathit{SELF}\text{:}\mathit{Ind}\right] \\ \text{cnt=}\left[\text{e:conductor(dudamel)}\right]\text{:}\mathit{RecType} \\ \text{c}_{\text{cnt}}\text{:content(e,cnt)}\end{array}\right] \text{] :[}\mathit{RecType}\text{]}\right] \\ \text{shared:} & \left[\begin{array}{l}\text{latest-utterance:}\mathit{Nil}\\ \text{commitments=[]:}\mathit{RecType}\end{array}\right]\end{array}\right]$$

Figure 2

$$\left[\begin{array}{lll} \text{move} & : & \left[\begin{array}{l} \text{e:}\mathit{Assertion}\underset{\cdot}{\wedge} \left[\text{sp=}\mathit{SELF}\text{:}\mathit{Ind}\right] \\ \text{cnt=}\left[\text{e:conductor(Dudamel)}\right]\text{:}\mathit{RecType} \\ \text{c}_{\text{cnt}}\text{:content(e,cnt)}\end{array}\right] \\ \text{chart} & : & \mathit{Chart} \\ \text{e} & : & \text{m-interp(chart,move)}\end{array}\right]$$

Figure 3

$$\left[\begin{array}{ll} \text{private:} & \left[\text{agenda=[} \left[\begin{array}{l} \text{e:}\mathit{Acknowledgement}\underset{\cdot}{\wedge} \left[\begin{array}{l}\text{sp=}u_1\text{.move.e.au:}\mathit{Ind}\\ \text{au=}\mathit{SELF}\text{:}\mathit{Ind}\end{array}\right] \\ \text{cnt=}u_1\text{.move.cnt:}\mathit{RecType} \\ \text{c}_{\text{cnt}}\text{:content(e,cnt)}\end{array}\right] \text{]:[}\mathit{RecType}\text{]}\right] \\ \text{shared:} & \left[\text{latest-utterance:} \left[\begin{array}{l}\text{move=}u_1\text{.move:}\mathit{Move}\\ \text{chart=}u_1\text{.chart:}\mathit{Chart}\\ \text{e=}u_1\text{.e:m-interp(chart,move)}\end{array}\right]\right]\end{array}\right]$$

Figure 4

$T_1\boxed{\underset{\cdot}{\wedge}}T_2$. If one or both of T_1 and T_2 are non-record types then $T_1\boxed{\underset{\cdot}{\wedge}}T_2$ will be T_2. If they are both record types, then for any label ℓ which occurs in both T_1 and T_2, $T_1\boxed{\underset{\cdot}{\wedge}}T_2$ will contain a field labelled ℓ with the type resulting from the asymmetric merge of the corresponding types in the ℓ-fields of the two types (in order). For labels which do not occur in both types, $T_1\boxed{\underset{\cdot}{\wedge}}T_2$ will contain the fields from T_1 and T_2 unchanged. In this informal statement we have ignored complications that arise concerning dependent types in record types. Our notion of asymmetric merge is related to the notion of priority unification (Shieber, 1986).

3 Phrase structure rules as update functions

We take signs to be records of the type (5).

(5) $$\left[\begin{array}{lll}\text{s-event} & : & \mathit{SEvent}\\ \text{cnt} & : & \mathit{Cnt}\end{array}\right]$$

This represents the pairing of a speech event with content in a Saussurean sign. It does not, however, require the presence of any hierarchical information in the sign corresponding to what in linguistic theory is normally referred to as the *constituent* (or *phrase*) structure of the utterance. To some extent it is arbitrary where we add this information. We could, for example, add it under the label 's-event' ("speech event"). However, it will be more convenient (in terms of keeping paths that we need to refer to often shorter) to add a third field labelled 'syn' ("syntax") at the top level of the sign type as in (6).

(6) $$\left[\begin{array}{lll}\text{s-event} & : & \mathit{SEvent}\\ \text{syn} & : & \mathit{Syn}\\ \text{cnt} & : & \mathit{Cnt}\end{array}\right]$$

However, as we will see below, *Syn* will require a 'daughters'-field for a string of signs. This means that *Sign* becomes a recursive type. It will be a *basic* type with its witnesses defined by (7).

(7) $$\sigma : \mathit{Sign} \text{ iff } \sigma : \left[\begin{array}{lll}\text{s-event} & : & \mathit{SEvent}\\ \text{syn} & : & \mathit{Syn}\\ \text{cnt} & : & \mathit{Cnt}\end{array}\right]$$

We shall take *Syn* to be the type (8).

(8) $\begin{bmatrix} \text{cat} & : & \mathit{Cat} \\ \text{daughters} & : & \mathit{Sign}^* \end{bmatrix}$

The type *Sign*, as so far defined, can be seen as a *universal resource*. By this we mean that it is a type which is available for all languages. *Cat* is the type of names of syntactic categories. For the purposes of the current toy example we will take the witnesses of *Cat* to be: s ("sentence"), np ("noun phrase"), det ("determiner"), n ("noun"), v ("verb") and vp ("verb phrase"). We will use capitalized versions of these category names to represent types of signs with the appropriate path in a sign type as in (9).

(9) a. $S \equiv \mathit{Sign} \mathbin{\dot\wedge} \left[\text{syn:}\left[\text{cat=s:}\mathit{Cat}\right]\right]$

b. $\mathit{NP} \equiv \mathit{Sign} \mathbin{\dot\wedge} \left[\text{syn:}\left[\text{cat=np:}\mathit{Cat}\right]\right]$

c. $\mathit{Det} \equiv \mathit{Sign} \mathbin{\dot\wedge} \left[\text{syn:}\left[\text{cat=det:}\mathit{Cat}\right]\right]$

d. $N \equiv \mathit{Sign} \mathbin{\dot\wedge} \left[\text{syn:}\left[\text{cat=n:}\mathit{Cat}\right]\right]$

e. $V \equiv \mathit{Sign} \mathbin{\dot\wedge} \left[\text{syn:}\left[\text{cat=v:}\mathit{Cat}\right]\right]$

f. $\mathit{VP} \equiv \mathit{Sign} \mathbin{\dot\wedge} \left[\text{syn:}\left[\text{cat=vp:}\mathit{Cat}\right]\right]$

This means that, for example, (9a) is the type in Figure 5.

We might think that the type *Cat* is a language specific resource and indeed if we were being more precise we might introduce separate types for different languages such as Cat_{eng}, Cat_{swe} and Cat_{tag} for the type of category names of English, Swedish and Tagalog respectively. However, there is a strong intuition that categories in different languages are more or less related. For example, we would not be surprised to find that the categories available for English and Swedish closely overlap (despite the fact that their internal syntactic structure differs) whereas the categories of English and Tagalog have less overlap. (See (Gil, 2000) for discussion.) For this reason we assume that there is a universal resource *Cat* and that each language will have a subtype of *Cat* which specifies which of the categories are used in that particular language. This is related to the kind of view of linguistic universals as a kind of toolbox from which languages can choose which is put forward by Jackendoff (2002) and Cooper and Ranta (2008).

The ontological status of objects of type *Cat* as we have presented them is a little suspicious. Intuitively, categories should be subtypes of *Sign*, as in (9). We have identified signs belonging to these types as containing a particular object in *Cat* in their 'cat'-field. But one might try to characterize such signs in a different way, for example, as fulfilling certain conditions such as having certain kinds of daughters. However, this is not quite enough, for example, for lexical categories, which do not have daughters. We have to have a way of assigning categories to words and we need to create something in the sign-type that will indicate the arbitrary assignment of a category to a word. For want of a better solution we will introduce the category names which belong to the type *Cat* as a kind of "book-keeping" device that will identify a sign-type as being one whose witnesses belong to category bearing that name.

The 'daughters'-field is required to be a string of signs, possibly the empty string, since the type *Sign** uses the Kleene-*, that is the type of strings of signs including the empty string, ε. Lexical items, that is words and phrases which are entered in the lexicon, will be related to signs which have the empty string of daughters. We will use *NoDaughters* to represent the type $\left[\text{syn:}\left[\text{daughters=}\varepsilon\text{:}\mathit{Sign}^*\right]\right]$.

If T_{phon} is a type (normally a phonological type, that is, $T_{\text{phon}} \sqsubseteq \mathit{Phon}$) and T_{sign} is a type (normally a sign type, that is,$T_{\text{sign}} \sqsubseteq \mathit{Sign}$, then we shall use Lex(T_{phon}, T_{sign}) to represent Figure 6. This means, for example, that Figure 7(a) represents the type in Figure 7(b) which, after spelling out the abbreviations, can be seen to be the type in Figure 7(c). We can think of 'Lex' as the function in (10)[3]

(10) $\lambda T_1\text{:}\mathit{Type}$
$\lambda T_2\text{:}\mathit{Type}$.
$T_1 \mathbin{\dot\wedge} \left[\text{s-event:}\left[\text{e:}T_2\right]\right] \mathbin{\dot\wedge} \mathit{NoDaughters}$

This function, which creates sign types for lexical items in a language, associating types with a syntactic category, can be seen as a universal resource. We can think of it as representing a (somewhat uninteresting, but nevertheless true) linguistic universal: "There can be speech events of given types which have no daughters (lexical items)".

[3]We are using the notational convention for function application as used, for example, by (Montague, 1973) that if f is a function $f(a, b)$ is $f(b)(a)$.

$$\left[\begin{array}{lll}
\text{s-event} & : & \left[\begin{array}{lll}
\text{e-loc} & : & \textit{Loc} \\
\text{sp} & : & \textit{Ind} \\
\text{au} & : & \textit{Ind} \\
\text{e} & : & \textit{Phon} \\
\text{c}_{\text{loc}} & : & \text{loc(e,e-loc)} \\
\text{c}_{\text{sp}} & : & \text{speaker(e,sp)} \\
\text{c}_{\text{au}} & : & \text{audience(e,au)}
\end{array}\right] \\
\text{syn} & : & \left[\begin{array}{lll}
\text{cat=s} & : & \textit{Cat} \\
\text{daughters} & : & \textit{Sign}^{*}
\end{array}\right] \\
\text{cnt} & : & \textit{Cnt}
\end{array}\right]$$

Figure 5

$$T_{\text{sign}} \underset{.}{\wedge} \left[\text{s-event:}\left[\text{e:}T_{\text{phon}}\right]\right] \underset{.}{\wedge} \textit{NoDaughters}$$

Figure 6

a. Lex(“Dudamel”, *NP*)

b. $\textit{NP} \underset{.}{\wedge} \left[\text{s-event:}\left[\text{e:“Dudamel”}\right]\right] \underset{.}{\wedge} \textit{NoDaughters}$

c.
$$\left[\begin{array}{lll}
\text{s-event} & : & \left[\begin{array}{lll}
\text{e-loc} & : & \textit{Loc} \\
\text{sp} & : & \textit{Ind} \\
\text{au} & : & \textit{Ind} \\
\text{e} & : & \text{“Dudamel”} \\
\text{c}_{\text{loc}} & : & \text{loc(e,e-loc)} \\
\text{c}_{\text{sp}} & : & \text{speaker(e,sp)} \\
\text{c}_{\text{au}} & : & \text{audience(e,au)}
\end{array}\right] \\
\text{syn} & : & \left[\begin{array}{lll}
\text{cat=np} & : & \textit{Cat} \\
\text{daughters=}\varepsilon & : & \textit{Sign}^{*}
\end{array}\right] \\
\text{cnt} & : & \textit{Cnt}
\end{array}\right]$$

Figure 7

The lexical resources needed to cover our example fragment is given in (11).

(11) Lex(“Dudamel”, *NP*)
Lex(“Beethoven”, *NP*)
Lex(“a”, *Det*)
Lex(“composer”, *N*)
Lex(“conductor”, *N*)
Lex(“is”, *V*)
Lex(“ok”, *S*)
Lex(“aha”, *S*)

The types in (11) belong to the specific resources required for English. This is not to say that these resources cannot be shared with other languages. Proper names like *Dudamel* and *Beethoven* have a special status in that they can be reused in any language, though often in modified form, at least in terms of the phonological type with which they are associated without this being perceived as quotation, code-switching or simply showing off that you know another language.

Resources like (11) can be exploited by update rules. If Lex(T_w, C) is one of the lexical resources available to an agent A and A judges an event e to be of type T_w, then A is licensed to update their gameboard with the type Lex(T_w, C). Intuitively, this means that if the agent hears an utterance of the word “composer”, then they can conclude that they have heard a sign which has the category noun. This is the beginning of *parsing*. The licensing condition corresponding to lexical resources like (11) is given in Figure 8. We will return below to how this relates to gameboard update. Figure 8 says that an agent with lexical resource Lex(T, C) who judges a speech event, u, to be of type T is licensed to judge that there is a sign of type Lex(T, C) whose 's-event.e'-field contains u.

Strings of utterances of words can be classified as utterances of phrases. That is, speech events are hierarchically organized into types of speech events. Agents have resources which allow them to reclassify a string of signs of certain types ("the daughters") into a single sign of another type ("the mother"). So for example a string of type $\mathit{Det}^\frown \mathit{N}$ (that is, a concatenation of an event of type *Det* and an event of type *N*) can lead us to the conclusion that we have observed a sign of type *NP* whose daughters are of the type $\mathit{Det}^\frown \mathit{N}$. The resource that allows us to do this is a rule which we will model as the function in (12a) which we will represent as (12b).

(12) a. $\lambda u : \mathit{Det}^\frown \mathit{N}\ .$
$$\mathit{NP} \mathbin{\dot\wedge} \left[\text{syn:}\left[\text{daughters=}u\text{:}\mathit{Det}^\frown \mathit{N}\right]\right]$$

b. RuleDaughters(*NP*, $\mathit{Det}^\frown \mathit{N}$)

'RuleDaughters' is to be the function in Figure 9. Thus 'RuleDaughters', if provided with a subtype of Sign^+ and a subtype of *Sign* as arguments, will return a function which maps a string of signs of the first type to the second type with the restriction that the daughters field is filled by the string of signs. 'RuleDaughters' is one of a number of sign type construction operations which we will introduce as universal resources which have the property of returning what we will call a sign combination function. The licencing conditions associated with sign combination functions are as characterized in Figure 10. This means, for example, that if you categorize a string of signs, u, as being of type $\mathit{Det}^\frown \mathit{N}$ then you can conclude that there is a sign of type *NP* with the additional restriction that its daughters are u.

'RuleDaughters' takes care of the 'daughters'-field but it says nothing about the 's-event.e'-field, that is the phonological type associated with the new sign. This should be required to be the concatenation of all the 's-event.e'-fields in the daughters. If $u : T^+$ where T is a record type containing the path π, we will use $\text{concat}_i(u[i].\pi)$, the concatenation of all the values $u[i].\pi$ for each element in the string u in the order in which they occur in the string. We can now formulate the function ConcatPhon as in Figure 11. ConcatPhon will map any string of speech events to the type of a single speech event whose phonology (that is the value of 's-event.e') is the concatenation of the phonologies of the individual speech events in the string.

We want to combine the function in Figure 11 with a function like that in (12). We do this by merging the domain types of the two functions and also merging the types that they return. This is shown in Figure 12(a) which in deference to standard linguistic notation for phrase structure rules could be represented as Figure 12(b).[4] In general we say that if $C, C_1, \ldots, C_n$ are category sign types as in (9) then $C \longrightarrow C_1 \ldots C_n$ represents RuleDaughters(C, $C_1{}^\frown \ldots {}^\frown C_n$) $\dot\wedge$ ConcatPhon where for any type returning functions $\lambda r : T_1\ .\ T_2(r)$ and $\lambda r : T_3\ .\ T_4(r)$ $\lambda r : T_1\ .\ T_2(r) \mathbin{\dot\wedge} \lambda r : T_3\ .\ T_4(r)$ denotes the function $\lambda r : T_1 \dot\wedge T_3\ .\ T_2(r) \dot\wedge T_4(r)$. Thus the function in Figure 12 can be represented in a third way as in Figure 13. The hope is that the ability to factorize rules into "bite-size" components will enable us to build a theory of resources that will allow us to study them in isolation and will also facilitate the development of theories of learning. It gives us a clue to how agents can build new rules by combining existing components in novel ways. It has implications for universality as well. For example, while the rule *NP* $\longrightarrow$ *Det N* is not universal (though it may be shared by a large number of languages), ConcatPhon is a universally available rule component, albeit a trivial universal, which says that you can have concatenations of speech events to make a larger speech event.

The rules associated with our small grammar are given by (13).

(13) *S* $\longrightarrow$ *NP VP*
NP $\longrightarrow$ *Det N*
VP $\longrightarrow$ *V NP*

4 Conclusions

It may seem that we have done an awful lot of work to arrive at simple phrase structure rules. Some readers might wonder why it is worth all this trouble to ground the rules in a theory of events

[4]Note that '$\longrightarrow$' used in the phrase structure rule in Figure 12(b) is not the same arrow as '$\rightarrow$' which is used in our notation for function types. We trust that the different contexts in which they occur will help to distinguish them.

If Lex(T, C) is a resource available to agent A, then for any u, $u :_A T$ licenses $:_A$ Lex(T, C) $\underset{.}{\wedge} \left[\text{s-event:} \left[\text{e}{=}u{:}T_1\right]\right]$

Figure 8

$$
\begin{array}{l}
\lambda T_1 : \mathit{Type} \\
\quad \lambda T_2 : \mathit{Type}\ . \\
\qquad \lambda u : T_1\ .\ T_2 \underset{.}{\wedge} \left[\text{syn:} \left[\text{daughters}{=}u{:}T_1\right]\right]
\end{array}
$$

Figure 9

If $f : (T_1 \rightarrow Type)$ is a sign combination function available to agent A, then for any u, $u :_A T_1$ licenses $:_A f(u)$

Figure 10

$$
\begin{array}{l}
\lambda u{:}\left[\text{s-event:}\left[\text{e:}\mathit{Phon}\right]\right]^{+}\ . \\
\quad \left[\ \text{s-event} \quad : \quad \left[\ \text{e}{=}\text{concat}_i(u[i].\text{s-event.e}) \quad : \quad \mathit{Phon}\ \right]\ \right]
\end{array}
$$

Figure 11

and action when what we come up with in the end is something that can be expressed in a standard notation which is one of the first things that a student of syntax learns. One reason has to do with our desire to explore the relationship between the perception and processing of non-linguistic events and speech events. Another reason has to do with placing natural constraints on syntax. By grounding syntactic structure in types of events we provide a motivation for the kind of discussion in (Cooper, 1982). An abstract syntax which proposes constituent structure which does not correspond to speech events is not grounded in the same way and thus presents a different kind of theory. The abstraction lies in the nature of the types used to classify strings, rather than abstract elements in the strings themselves. A third reason is that it points to a way of thinking of parsing in TTR as incremental updating of an information state similar to the kind of proposals that have been made in PTT (Poesio and Traum (1997) and Poesio and Rieser (2010)). We have not integrated our view of syntax with compositional semantics and dialogue update rules here. This is, however, done in the work in progress cited in footnote 1.

Acknowledgments

This paper was supported in part by VR project 2009-1569, Semantic analysis of interaction and coordination in dialogue (SAICD). I am grateful to three anonymous referees for insightful comments.

a. $\lambda u : \mathit{Det}\frown\mathit{N} \underset{.}{\wedge} \left[\text{s-event:}\left[\text{e:}\mathit{Phon}\right]\right]^{+}\ .$
$$
\begin{array}{l}
\quad \mathit{NP} \underset{.}{\wedge} \left[\text{syn:}\left[\text{daughters}{=}u{:}\mathit{Det}\frown\mathit{N}\right]\right] \\
\qquad \underset{.}{\wedge} \left[\text{s-event:}\left[\text{e}{=}\text{concat}_i(u[i].\text{s-event.e}){:}\mathit{Phon}\right]\right]
\end{array}
$$

b. $\mathit{NP} \longrightarrow \mathit{Det}\ \mathit{N}$

Figure 12

RuleDaughters(*NP*, $\mathit{Det}\frown\mathit{N}$) $\underset{:}{\wedge}$ ConcatPhon

Figure 13

References

Robin Cooper and Aarne Ranta. 2008. Natural Languages as Collections of Resources. In Robin Cooper and Ruth Kempson, editors, *Language in Flux: Dialogue Coordination, Language Variation, Change and Evolution*, volume 1 of *Communication, Mind and Language*, pages 109–120. College Publications, London.

Robin Cooper. 1982. Binding in wholewheat* syntax (*unenriched with inaudibilia). In Pauline Jacobson and Geoffrey K. Pullum, editors, *The Nature of Syntactic Representation*, volume 15 of *Synthese Language Library*. Reidel Publishing Company.

Robin Cooper. 2012. Type theory and semantics in flux. In Ruth Kempson, Nicholas Asher, and Tim Fernando, editors, *Handbook of the Philosophy of Science*, volume 14: Philosophy of Linguistics, pages 271–323. Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Vera Demberg, Frank Keller, and Alexander Koller. 2013. Incremental, predictive parsing with psycholinguistically motivated tree-adjoining grammar. *Computational Linguistics*, 39(4):1025–1066.

Arash Eshghi, Julian Hough, Matthew Purver, Ruth Kempson, and Eleni Gregoromichelaki. 2012. Conversational Interactions: Capturing Dialogue Dynamics. In Staffan Larsson and Lars Borin, editors, *From Quantfication to Conversation: Festschrift for Robin Cooper on the occasion of his 65th birthday*, volume 19 of *Tributes*, pages 325–349. College Publications.

David Gil. 2000. Syntactic categories, cross-linguistic variation and universal grammar. In Petra M. Vogel and Bernard Comrie, editors, *Approaches to the typology of word classes*, volume 23 of *Empirical approaches to language typology*. Mouton de Gruyter, Berlin.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press, Oxford.

Ray Jackendoff. 2002. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, University of Gothenburg.

J. McCarthy and P. J. Hayes. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502.

Richard Montague. 1973. 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.

M. Poesio and H. Rieser. 2010. Completions, coordination, and alignment in dialogue. *Dialogue and Discourse*, 1(1):1–89.

M. Poesio and D. Traum. 1997. Conversational actions and discourse situations. *Computational Intelligence*, 13(3):309–347.

Matthew Purver, Eleni Gregoromichelaki, Wilfried Meyer-Viol, and Ronnie Cann. 2010. Splitting the *I*s and Crossing the *You*s: Context, Speech Acts and Grammar. 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*, pages 43–50, Poznań. Polish Society for Cognitive Science.

Murray Shanahan. 2009. The frame problem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. `http://plato.stanford.edu/archives/win2009/entries/frame-problem/`, winter 2009 edition.

Stuart Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Publications, Stanford.