Formal semantics for perceptual classification
PREPRINT VERSION

Staffan Larsson
Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg
Box 200, SE40530 Göteborg, Sweden
sl@ling.gu.se
Telephone: +46 786 4378

Abstract
A formal semantics for low-level perceptual aspects of meaning is presented, tying these together
with the logical-inferential aspects of meaning traditionally studied in formal semantics. The key idea is
to model perceptual meanings as classifiers of perceptual input. Furthermore, we show how perceptual
aspects of meaning can be updated as a result of observing language use in interaction, thereby enabling
fine-grained semantic plasticity and semantic coordination. This requires a framework where intensions
are (1) represented independently of extensions, and (2) structured objects which can be modified as a
result of learning. We use Type Theory with Records (TTR), a formal semantics framework which starts
from the idea that information and meaning is founded on our ability to perceive and classify the world,
i.e., to perceive objects and situations as being of types. As an example of our approach, we show how a
simple classifier of spatial information based on the Perceptron can be cast in TTR.

Keywords: Type Theoretic Semantics, Perception, Statistical Classifiers, Learning, Dialogue

This is a preprint of an article published in Journal of Logic and Computation 2013. Published by Oxford
University Press. All rights reserved. Doi: 10.1093/logcom/ext059

1 Introduction

How is linguistic meaning related to perception? How do we learn and agree on the meanings of our words?
If a speaker of English is unable to distinguish gloves from mittens, most people would probably agree that
something is missing in this person’s knowledge of the meaning of “glove”. Similarly, if we tell A to find
some nice pictures of dogs chasing cats, and A comes back happily with an assortment of pictures displaying
lions chasing zebras, we would question whether A really knows the full meaning of the words “dog” and
“cat”; perhaps A is not a fully competent speaker of English2. If we found out, e.g., that A is not a native
speaker of English, we may take this to explain her odd take on the meaning of “cat” and “dog”. Part of
learning a language, it seems, is learning to identify individuals and situations that are in the extension of
the phrases and sentences of the language. For many concrete expressions, this identification relies crucially
on the ability to perceive the world, and to use perceptual information to classify individuals and situations
as falling under a given linguistic description or not. This view is consistent with several theories of word
meaning as grounded in sensory (or embodied) representations which have emerged during the last decade or

1http://logcom.oxfordjournals.org/cgi/content/abstract/ext059?ijkey=4Zz9vROsbXn3ZMy&keytype=ref
2These simple examples are of course subject to a number of preconditions, e.g., that A does not have serious problems with her
eyesight.
so (Barsalou et al., 2003; Roy, 2005; Steels and Belpaeme, 2005; Kelleher et al., 2005; Skočaj et al., 2010).
The idea is, roughly, that meanings of e.g. colour terms involve the same representations as are used in the perception of colours.

The problem we will be addressing here is how perceptual meanings and low-level perceptual data can be integrated into formal semantics. Perceptual meaning is an important aspect of the meaning of linguistic expressions referring to physical objects (such as concrete nouns or noun phrases). Knowing the perceptual meaning of an expression allows an agent to identify perceived objects and situations falling under the meaning of the expression. For example, knowing the perceptual meaning of “blue” would allow an agent to correctly identify blue objects. Similarly, an agent which is able to compute the perceptual meaning of “a boy hugs a dog” will be able to correctly classify situations where a boy hugs a dog. By integrating perceptual meanings and low-level perceptual data into formal semantics, we will enable mixing low-level (perceptual) and high-level (logical-inferential) meaning in a single framework. One may perhaps compare this with the idea in HPSG of mixing syntax and semantics in a single system (Pollard and Sag, 1994).

The last decades has seen the growth of an extensive body of work on computational models of perception and classification of perceptual (and other non-logical) information, mostly in the area of computational learning theory. An essential notion here is that of the (statistical) classifier, a computational device determining what class an item belongs to, based on various properties of the item. Crucially, these properties need not be encoded in some high-level representation language (such as logic or natural language). Instead, it may consist entirely of numeric data encoding more or less “low-level” information about the item in question, for example perceptual data. Perceptual data from sensory organs is often represented computationally as a vectors of real, integer, or binary numbers. For example, the output from a digital camera can be represented as a large vector of integers. From such very basic low-level data, slightly more refined but still low-level information can be extracted, i.e. contours (using edge detection algorithms), colour parameters (using colour encoding algorithms) and positions (using object tracking algorithms) can be extracted. The role of the classifier is to take this more or less refined low-level information derived from some entity (object, relation, situation) and to output a high-level result in the form of a judgement about the class (or type) of the entity. There are already computational models of neural networks acting as classifiers connected to linguistic expressions (colour terms), including learning of perceptual aspects of meaning (Steels and Belpaeme, 2005), but these models do not attempt to connect to the formal semantics tradition.

Classifiers can of course be implemented in many ways. However, most if not all can be defined formally as mathematical functions. Typically, the domain of a classifier function is numerical (e.g. real-valued, integer or binary) vectors and the range is a set of classes (or in the case of binary classifiers, equivalents of “yes” and “no”). The crucial step in making use of classifiers in formal semantics is to regard them as (parts of) representations of intensions of linguistic expressions. Traditionally, the intension of an expression helps determine whether some item belongs to the extension of the expression. Here, this translates to using a classifier to help determine whether some perceptual data derived from some item can be used to classify that item as falling under the expression, i.e., to be included in its extension.\(^3\)

The purpose of the present paper is to account for both intensions as classifiers in a framework which also encompasses accounts of many problems traditionally studied in formal semantics (such as inference, quantification, modality, etc.). To this end, we will be using Type Theory with Records, or TTR. As many other type theories, TTR is based on the notion of judgements of entities being of certain types – for example, a judgement that a certain situation is of a certain type. TTR starts from the idea that information and meaning is founded on our ability to perceive and classify the world, i.e., to perceive objects and situations as being

\(^3\)The idea of regarding classifiers as representing intensions is related to proposals by Muskens (2005) and Lappin (2012) to identify the intensions of an expression with an algorithm (implemented using logic programming or a functional programming language, respectively) for determining its extension. This paper can perhaps be regarded as an application of this general idea to the issue of perceptual meaning (although we are here using general notation for functions rather than any specific programming language).
of types.

The notion of a judgement is also connected to the notion of an agent making the judgement. Throughout, we will be modelling not meanings and situations per se, but only agents’ takes on meanings and situations. This may appear to be a solipsistic view which fails to account for the fact that we as humans (or at least as members of linguistic communities) share a world and a language. However, we take an essential part of the account of linguistic meaning to be a description of how agents coordinate their takes on the world and on the meanings of their linguistic expressions. The essential vehicle for this coordination is dialogue, or more generally linguistic interaction. By interacting with each other, agents reciprocally learn from each other and thereby come to have more or less coordinated (shared) takes on the world and on language. This view is what allows us to connect perception to meaning. Perception, it seems, is inherently agent-specific – there is no notion of objective perception – so if we are to include perception in semantics we need to somehow explain how individual perception relates to public meaning. They key to this is semantic coordination, i.e. the process of interactively coordinating on the meanings of linguistic expressions. Interactive coordination and reciprocal learning require semantic plasticity, i.e. the ability to modify meanings. A requirement on our semantics is therefore that it enables the kinds of modifications needed to account for semantic coordination of perceptual meanings.

The work presented here is part of a research program whose aim is to provide a formal semantics which is able to account for semantic coordination and semantic plasticity. The fact that language changes both in the long term and in the short term (even in the course of a single dialogue) has received increased interest from formal linguists in recent years (Cooper and Kempson, 2008). In recent work on vagueness, classical model theoretic semantics has been extended to deal with some aspects of semantic plasticity (Barker, 2002; Lassiter, 2011). In previous work, we have developed an account of semantic coordination and semantic plasticity (Cooper and Larsson, 2009; Larsson and Cooper, 2009; Larsson, 2009, 2010; Fernández et al., 2011), focusing on what we here refer to has logical-inferential meaning. From the viewpoint of this research programme, the purpose of the present work is to extend the account of semantic coordination and plasticity to cover perceptual aspects of meaning.

The structure of this paper is as follows. We will first introduce a simple language game which we will be using to illustrate our points and make them more concrete. We will then briefly introduce the TTR framework. In the following section, we show how perceptrons can be represented in TTR and how this can be used for incorporating subsymbolic semantics into a dynamic semantic / information state update framework. We will the discuss the notion of semantic coordination and how this contributes to a view of meaning as essentially social (or interpersonal).

2 The left-or-right game

To make our point as clearly as possible, we will show how an extremely simple kind of low-level perceptual data, namely position as encoded by a two-dimensional real-valued vector, can be integrated into a formal semantics framework. We are thus not trying to advance the state of the art in the processing of low-level perceptual data, nor the computational modelling of perceptual features. However, we believe that our account of how perceptual meanings into formal semantics can be generalised to more advanced models of low-level perceptual input.

As an illustration, we will be using a simple language game whose objective is to negotiate the meanings of the words “left” and “right”\(^4\). A and B are facing a framed surface on a wall, and A has a bag of objects which can be attached to the framed surface. The following procedure is repeated:

1. A places an object in the frame

\(^4\)Actually, in our examples we will only be concerned with the word “right".
2. B orients to the new object, assigns it a unique individual marker and orients to it as the current object in shared focus of attention

3. A says either “left” or “right”

4. B interprets A’s utterance based on B’s take on the situation. Interpretation involves determining whether B’s understanding of A’s utterance is consistent with B’s take on the situation.

5. If an inconsistency results from interpretation, B assumes A is right, says “aha”, and learns from this exchange; otherwise, B says “okay”

Note that the resulting meanings of “left” and “right” will depend on how A places the objects in the frame and what A says when doing so; this may or may not correspond to the standard everyday meanings of “left” and “right”. However, to keep things intuitive we will assume that A’s takes on the meanings of these words can be paraphrased as “to the left of the center of the frame” and “to the right of the center of the frame”, respectively.

Below is a sample interaction from the left-or-right game, consisting of two rounds. The most recent object placed in the frame by A is represented as a dot; previously placed objects are represented by circles.

(1) A places first object:

A: “right”
B: “okay”

A places second object:

A: “right”
B: “aha”

Not also that B, if faced with an inconsistency between B’s take on the situation and (B’s understanding of) A’s utterance, will always yield to A’s judgement and adapt to it. In general, however, an agent faced with an inconsistency derived from contextual interpretation may react in a number of different ways including explicit rejection of (some or all of) the utterance in question, issuing a clarification request, or providing some other kind of feedback to indicate that there is a problem.

The left-or-right game can be regarded as a considerably pared-down version of the “guessing game” in Steels and Belpaeme (2005), where perceptually grounded colour terms are learnt from interaction. The kinds of meanings learnt in the left-or-right game may be considered trivial. However, at the moment we are mainly interested in the basic principles of combining formal dynamic semantics with learning of perceptual meaning from dialogue, and the hope is that these can be formulated in a general way which can later be used in more interesting settings.
3 Dynamic semantics in TTR

We argue that TTR is well suited for dealing with the problems we are interested in. In TTR, types are first-class objects, which allows perceptual classifier functions to be formalised and used in representing meanings of linguistic expressions together with the high-level aspects of meaning traditionally studied in formal semantics. Furthermore, TTR integrates logical techniques such as binding and the lambda-calculus into feature-structure like objects called record types. Thus we get more structure than in a traditional formal semantics and more logic than is available in traditional unification-based systems. The feature structure like properties are important for developing similarity metrics on meanings and for the straightforward definition of meaning modifications involving refinement and generalization. The logical aspects are important for relating our semantics to the model and proof theoretic tradition associated with compositional semantics. Semantic phenomena which have been described using TTR include modelling of intensionality and mental attitudes (Cooper, 2005), dynamic generalised quantifiers (Cooper, 2004), co-predication and dot types in lexical innovation, frame semantics for temporal reasoning, reasoning in hypothetical contexts (Cooper, 2011), enthymematic reasoning (Breitholtz and Cooper, 2011), clarification requests (Cooper, 2010), negation (Cooper and Ginzburg, 2011), and information states in dialogue (Cooper, 1998; Ginzburg, 2012).

3.1 TTR: A brief introduction

We can here only give a brief and partial introduction to TTR; see also Cooper (2005) and Cooper (2012). To begin with, \( a : T \) means that \( a \) is of type \( T \). One basic type in TTR is \( \text{Ind} \), the type of an individual; another basic type is \( \text{Real} \), the type of real numbers. Given that \( T_1 \) and \( T_2 \) are types, \( T_1 \rightarrow T_2 \) is a functional type whose domain is objects of type \( T_1 \) and whose range is objects of type \( T_2 \). Next, we introduce records and record types. If \( a_1 : T_1, a_2 : T_2(a_1), \ldots, a_n : T_n(a_1, a_2, \ldots, a_{n-1}) \), the record to the left is of the record type to the right:

\[
\begin{array}{c}
\ell_1 = a_1 \\
\ell_2 = a_2 \\
\vdots \\
\ell_n = a_n \\
\vdots
\end{array}
\quad:
\begin{array}{c}
\ell_1 : T_1 \\
\ell_2 : T_2(\ell_1) \\
\vdots \\
\ell_n : T_n(\ell_1, \ell_2, \ldots, \ell_{n-1})
\end{array}
\]

In (2), \( \ell_1, \ldots, \ell_n \) are labels which can be used elsewhere to refer to the values associated with them. A sample record and record type is shown in (3).

\[
\begin{array}{c}
\text{ref} = \text{obj}_{123} \\
\text{c}_{\text{man}} = \text{prf} \left( \text{man}(\text{obj}_{123}) \right) \\
\text{c}_{\text{run}} = \text{prf} \left( \text{run} \left( \text{obj}_{123} \right) \right)
\end{array}
\quad:
\begin{array}{c}
\text{ref} : \text{Ind} \\
\text{c}_{\text{man}} : \text{man} \left( \text{ref} \right) \\
\text{c}_{\text{run}} : \text{run} \left( \text{ref} \right)
\end{array}
\]

Records (and record types) can be nested, so that the value of a label is itself a record (or record type). As can be seen in (3), types can be constructed from predicates, e.g., “run” or “man”. Such types are called ptypes and correspond roughly to propositions in first order logic. A fundamental type-theoretical intuition is that something of a ptype \( P(a_1, \ldots, a_n) \) is whatever it is that counts as a proof of \( P(a_1, \ldots, a_n) \). One way of putting this is that “propositions are types of proofs”. In (3), we simply use \( \text{prf}(P) \) as a placeholder for proofs of \( P \); below, we will have more to say about what proofs can be \(^5\).
Types constructed with predicates may be dependent. This is represented by the fact that arguments to the predicate may be represented by labels used on the left of the ‘:’ elsewhere in the record type. In (3), the type of \( c_{\text{man}} \) is dependent on ref (as is \( c_{\text{run}} \)).

Some of our types will contain manifest fields (Coquand et al., 2004) like the \( c_{\text{man}} \)-field in (4):

\[
\begin{array}{c}
ref & : & \text{Ind} \\
{c_{\text{man}}} = \text{prf}_{23} & : & \text{man(ref)}
\end{array}
\]

In (4), \( c_{\text{man}} = \text{prf}_{23} : \text{man(ref)} \) is a convenient notation for \( c_{\text{man}} : \text{man(ref)}_{\text{prf}_{23}} \) where \( \text{man(ref)}_{\text{prf}_{23}} \) is a singleton type. If \( a : T \), then \( T_a \) is a singleton type and \( b : T_a \) iff \( b = a \). Manifest fields allow us to progressively specify what values are required for the fields in a type.

An important notion in this kind of type theory is that of subtype. Formally, \( T_1 \sqsubseteq T_2 \) means that \( T_1 \) is a subtype of \( T_2 \), which informally means that everything of type \( T_1 \) is also of type \( T_2 \). Two examples of subtyping are shown in (5).

\[
\begin{array}{c}
\text{ref} & : & \text{Ind} \\
c & : & \text{glove(ref)} \\
\text{ref} = \text{obj}_{123} & : & \text{Ind}
\end{array} \sqsubseteq \begin{array}{c}
\text{ref} & : & \text{Ind} \\
\end{array}
\]

First, if a record type \( T \) contains all the labels of \( T' \) and assign them the same types as \( T' \), but also some additional fields, then \( T \sqsubseteq T' \). Second, if \( T \) contains a manifest field \( \ell = a : T'' \) and \( T \) contains \( \ell : T'' \), then also \( T \sqsubseteq T' \). For our present purposes, these are the only relevant kinds of subtyping.

We will also have use for an operator for combining record types. Suppose that we have two record types \( C_1 \) and \( C_2 \):

\[
C_1 = \begin{array}{c}
\text{ref} : \text{Ind} \\
{c_{\text{man}}} : \text{man(ref)}
\end{array} \\
C_2 = \begin{array}{c}
\text{ref} : \text{Ind} \\
{c_{\text{run}}} : \text{run(ref)}
\end{array}
\]

In this case, \( C_1 \land C_2 \) is a type; more specifically, a meet type. In general if \( T_1 \) and \( T_2 \) are types then \( T_1 \land T_2 \) is a type and \( a : T_1 \land T_2 \) iff \( a : T_1 \) and \( a : T_2 \).

\[
C_1 \land C_2 = \begin{array}{c}
\text{ref} : \text{Ind} \\
{c_{\text{man}}} : \text{man(ref)}
\end{array} \land \begin{array}{c}
\text{ref} : \text{Ind} \\
{c_{\text{run}}} : \text{run(ref)}
\end{array}
\]

A meet type \( T_1 \land T_2 \) of two record types can be simplified to a new record type by a process similar to unification in feature-based systems. We will represent the simplified type by putting a dot under the symbol \( \land \). Thus if \( T_1 \) and \( T_2 \) are record types then there will be a type \( T_1 \land T_2 \) equivalent to \( T_1 \land T_2 \) (in the sense that something will be of the first type if and only if it is of the second type).

\[
C_1 \land C_2 = \begin{array}{c}
\text{ref} : \text{Ind} \\
{c_{\text{man}}} : \text{man(ref)} \\
{c_{\text{run}}} : \text{run(ref)}
\end{array}
\]

\[6\text{The TTR subtype relation is given a model-theoretic and a syntactic definition in Cooper (2008). An alternative syntactic definition is given in Cooper (2012).}\]
The operation \( \land \), referred to as *merge* below, corresponds to unification in feature-based systems and its definition (which we omit here, but see Cooper (2005)) is similar to the graph unification algorithm.

If \( P \) is a ptype, \( \neg P \) is the negation of the ptype \( P \) (however, \( \neg P \) is not itself a ptype). \( \neg P \) is equivalent to \( P \rightarrow \perp \) where \( \perp \) is a type that is always empty no matter what is assigned to the basic types. Finally, a record type \( RT \) which contains both a field with type \( T \) and a field with type \( \neg T \) is inconsistent \((RT \approx \perp)\). An example is shown in (9).

\[
\begin{pmatrix}
\text{ref} & : & \text{Ind} \\
\text{c}_{\text{run}} & : & \text{run}(\text{ref}) \\
\text{c}_{\neg \text{run}} & : & \neg \text{run}(\text{ref})
\end{pmatrix} \approx \perp
\]

If a record type is inconsistent, there can be no records of that type.

### 3.2 Static and dynamic semantics

In representing meanings, we follow Cooper (2005) in distinguish constraints on the context (roughly, presuppositions) from the content of an expression. We thus take parts of the meaning of an uttered expression to be *backgrounded*, and other parts to be *foregrounded*. The static meaning of an expression \( e \) is represented as a function from a context (represented by a record) to a content (represented by a record type).

\[
\lambda r : T_{bg}(T_{fg})
\]

Here, the background meaning \( T_{bg} \) is a record type representing constraints on the context (a record), and \( T_{fg} \) represents the information conveyed by the expression. As an example, the static meaning of “The man runs” would be represented as in (11).

\[
\lambda r : \begin{pmatrix}
\text{ref} & : & \text{Ind} \\
\text{c}_{\text{man}} & : & \text{man}(\text{ref}) \\
\text{c}_{\text{run}} & : & \text{run}(\text{r}.ref)
\end{pmatrix}
\]

We will be representing various aspects of the meaning of a linguistic expression \( e \) as a record specifying background meaning separately from the static meaning:

\[
\llbracket e \rrbracket = \begin{pmatrix}
\text{bg} & = & T_{bg} \\
\text{f} & = & \lambda r : \text{bg}(T_{fg})
\end{pmatrix}
\]

Representing \( T_{bg} \) separately is motivated by the need to check that the constraints on the context are fulfilled by the current take on the context prior to updating it (see below). Furthermore, this kind of representation, where some aspects of the meaning of an expression are separated out from the (static) meaning function, is useful in allowing various kinds of manipulation of meanings and hence for dealing with semantic plasticity and semantic coordination\(^7\). As an example, the sentence “The man runs” could be represented thus:

\[
\llbracket \text{“The man runs”} \rrbracket = \begin{pmatrix}
\text{bg} & = & \begin{pmatrix}
\text{ref} & : & \text{Ind} \\
\text{c}_{\text{man}} & : & \text{man}(\text{ref})
\end{pmatrix} \\
\text{f} & = & \lambda r : \text{bg}(\begin{pmatrix}
\text{c}_{\text{run}} & : & \text{run}(\text{r}.ref)
\end{pmatrix})
\end{pmatrix}
\]

\(^7\)Although embedded in a function, the foreground meaning can still be modified although in a slightly more complicated way using operations on functions such as the merge operator for functions defined in Cooper (2012).
To formulate the dynamic meaning (context update potential) of \( e \), we will employ the notion of a fixed point type (Cooper, 2005). If we have a function \( f = \lambda x : T_1(T_2) \), then \( a \) is a fixed point for this function just in case \( a : f(a) \). The fixed point type of \( f \), denoted \( \mathcal{F}(f) \), is obtained by extending the type of the domain of \( f \) with the dependent type that characterises its range. The fixed point type for the static meaning of “The man runs” is shown in (14).

\[
\mathcal{F}(\lfloor \text{The man runs} \rfloor) = \mathcal{F}(\lambda r:\text{Ind}: \begin{array}{c}
\text{c\_man} : \text{man}(r) \\
\text{c\_run} : \text{run}(r)
\end{array})
\]

In TTR, contexts are represented as records, whereas an agent’s takes on a context is represented as a record type (typically involving manifest fields). This allows takes on contexts to be underspecified, which is useful in modeling agents with incomplete knowledge.

From a dynamic semantics perspective, \( T_{fg} \) represents the information to be added to an agent’s take on the context by the expression in question. \( T_{bg} \) represents constraints on the agent’s take on the context prior to integrating the expression. However, as the take on the context is itself a record type, the constraint is now expressed by the subtyping relation. In (15) we characterise updating an agent’s take on the context with an expression \( e \), given a current take on the context \( s_t \).

\[
\text{If } s_t \sqsubseteq \lfloor e \rfloor_{\text{bg}} \text{ then } s_{t+1} = s_t \land \mathcal{F}(\lfloor e \rfloor_{\text{f}})
\]

For example, assume that an agent A’s take on the context is as \( s^A_1 \) in (16).

\[
s^A_1 = \begin{array}{c}
\text{ref} = \text{ind}123 : \text{Ind} \\
\text{c\_man} : \text{man}(\text{ref})
\end{array}
\]

This context includes a specific individual who is a man\(^9\). To update the context with an utterance of “The man runs”, A first verifies that \( s_1 \sqsubseteq \lfloor \text{“The man runs”} \rfloor_{\text{bg}} \):

\[
\begin{array}{c}
\text{ref} = \text{ind}123 : \text{Ind} \\
\text{c\_man} : \text{man}(\text{ref})
\end{array} \sqsubseteq \begin{array}{c}
\text{ref} : \text{Ind} \\
\text{c\_man} : \text{man}(\text{ref})
\end{array}
\]

Then, A computes \( s^A_2 \):

\[
\begin{array}{c}
\text{ref} = \text{ind}123 : \text{Ind} \\
\text{c\_man} : \text{man}(\text{ref})
\end{array} \land \mathcal{F}(\lfloor \text{“The man runs”} \rfloor_{\text{f}}) = \begin{array}{c}
\text{ref} : \text{Ind} \\
\text{c\_man} : \text{man}(\text{ref}) \\
\text{c\_run} : \text{run}(\text{ref})
\end{array}
\]

\(^8\)A function of this kind is called a “family of types” in Cooper (2005).

\(^9\)Proofs are not included in this example.
4 Dynamic semantics for perception

In this section, we will show how the TTR-based dynamic semantics presented above in Section 3 can be extended to incorporate low-level perceptual aspects of meaning. Examples will be based on the left-or-right game presented in Section 2.

4.1 Perceptual meanings as classifiers

We take the lexical meaning $⟦e⟧$ of an expression $e$ to often contain not only logical-inferential semantics but also perceptual meaning (at least for non-abstract expressions). By this we mean that aspect of the meaning of an expression which allows an agent to detect objects or situations referred to by the expression $e$. For example, knowing the perceptual meaning of “panda” allows an agent to correctly classify pandas in her environment as pandas. Likewise, an agent which is able to compute the perceptual meaning of “a boy hugs a dog” will be able to correctly classify situations where a boy hugs a dog. We can therefore think of perceptual meanings as classifiers of sensory input. Classification of perceptual input can be regarded as a mapping of sensor readings to ptypes or negations of ptypes.

4.2 The TTR perceptron

To represent perceptual classifiers, we will be using a simple perceptron (Rosenblatt, 1958). A perceptron is a very simple neuron-like object with several inputs and one output. Each input is multiplied by a weight and if the summed inputs exceed a threshold, the perceptron triggers, i.e. it yields 1 as output; otherwise it yields 0 (or in some versions -1).

\[
o(x) = \begin{cases} 
1 & \text{if } w \cdot x > t \\
0 & \text{otherwise}
\end{cases}
\]

where $w \cdot x = \sum_{i=1}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$

Perceptrons are limited to learning problems which are linearly separable; the distinction between left and right is one such problem\(^\text{10}\). Perceptrons can be interconnected by connecting the output of one or several perceptrons to the inputs of a different perceptron. Also, perceptrons can also be used to model reasoning. Here, we want to use a single perceptron to model perception.

In TTR, an $n$-dimensional real-valued vector will be represented as a record with labels $1, \ldots, n$ where the value of each label will be a real number. Such a records will be of the type RealVector$_n$.

\[
\text{RealVector}_n = \begin{bmatrix} 1 & \text{Real} \\
2 & \text{Real} \\
\vdots & \text{Real} \\
n & \text{Real} \end{bmatrix}
\]

\[
x = \begin{bmatrix} 1 = 0.23 \\
2 = 0.34 \\
3 = 0.45 \end{bmatrix} : \text{RealVector}_3
\]

\(^{10}\)This holds for these terms in their “literal”, purely geometrical sense. There are of course also a host of other uses of “left” and “right” which are not likely to be linearly separable. One example is the use of these terms in the sphere of politics. Also, since even a seemingly straightforward term like “above” may evidently involve functional aspects (Coventry, 1999), it seems likely that even some fairly “literal” uses of “left” and “right” can involve more than geometry.
For convenience, we will abbreviate this as in this example:

\[(21)\; \mathbf{x} = \begin{bmatrix} 0.23 & 0.34 & 0.45 \end{bmatrix} : \text{RealVector}_3\]

\[\mathbf{x}_n = \mathbf{x}.n, \text{ so } \mathbf{x}_2 = 0.34\]

We will also use a general type \text{RealVector} (without subscript) for real-valued vectors of any dimensionality.

\[(22)\; \mathbf{x} : \text{RealVector} \text{ if } \mathbf{x} : \text{RealVector}_n \text{ for some } n \geq 1\]

The basic perceptron returns a real-valued number (1.0 or 0.0) but when we use a perceptron as a classifier we want it to instead return a ptype, i.e. a type constructed from a predicate and some number of arguments, or the negation of a ptype. We also want the individual or individuals to be classified as input to the function. A TTR perceptron \(\pi_p\) for an \(n\)-ary predicate \(p\) is a function

\[(23)\; \pi_p : \text{RealVector} \rightarrow \text{Real} \rightarrow \left[ \begin{array}{c} \ell_1 : \text{Ind} \\ \vdots \\ \ell_n : \text{Ind} \\ sr : \text{RealVector} \end{array} \right] \rightarrow \text{Type}\]

such that

\[(24)\; \text{if } r : \left[ \begin{array}{c} \ell_1 : \text{Ind} \\ \vdots \\ \ell_n : \text{Ind} \\ sr : \text{RealVector} \end{array} \right], \text{ then } \pi_p(w,t)(r) = \begin{cases} p(r.\ell_1, \ldots, r.\ell_n) & \text{if } r \cdot w > t \\ \neg p(r.\ell_1, \ldots, r.\ell_n) & \text{otherwise} \end{cases}\]

5 \hspace{1em} Contextual interpretation in the left-or-right game

We will now show a detailed example of how the TTR perceptron is used in contextual interpretation of “right” in the left-or-right game. First, however, we need to make more concrete the notion of situation we are using here, and how it relates to an agent’s sensors. We also introduce the idea of sensor readings as proofs.

5.1 Situations, sensors and the focused object

In first language acquisition, training of perceptual classifiers typically takes place in situations where the referent is in the shared focus of attention and thus perceivable to the dialogue participants, and for the time being we limit our analysis to such cases. For our current purposes, we assume that our DPs (Dialogue Participants) are able to establish a shared focus of attention, and we will designate the label foc for the object or objects taken by a DP to be in shared focus.

A (simple) sensor collects some information (sensor input) from the environment and emits a real-valued vector. The sensor is assumed to be oriented towards the object in shared focus. An agent’s (possibly underspecified) take on a situation is a part of the agent’s information state. It is represented as a record type, possibly containing manifest fields. Furthermore, we will assume that sensors are directed towards the focused object.

In the left-or-right game, we will assume that B’s take on the situation includes readings from a position sensor (denoted “sr_{pos}”) and a field foc for an object in shared focus of attention. The position sensor returns
a two-dimensional real-valued vector representing the horizontal vertical coordinates of the focused object: 
\[
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]
where \(-1.0 \leq x, y \leq 1.0\) and \([0.0 \ 0.0]\) represents the centre of the frame.

Here is an example of B’s take on the situation prior to playing a round of the left-or-right game:

\[
\begin{align*}
(25) \quad s^B_1 = & \begin{bmatrix}
\text{sr}_\text{pos} = & [0.900 \ 0.100] : \text{RealVector} \\
\text{foc} = & \text{obj}_{45} : \text{Ind} \\
\text{spkr} = & \text{A} : \text{Ind}
\end{bmatrix}
\end{align*}
\]

In \(s^B_1\), B’s sensor is oriented towards \text{obj}_{45} and \text{sr}_\text{pos} returns a vector corresponding to the position of \text{obj}_{45}.

As mentioned above, something of the ptype \(p(a_1, \ldots, a_n)\) is whatever it is that counts as a proof of \(p(a_1, \ldots, a_n)\). One can have different ideas of what kind of objects count as proofs. Here we will be assuming that proof-objects can be records including readings from sensors. These records will also contain fields specifying the individuals \(a_1, \ldots, a_n\). The idea is that the sensor reading indicates that individuals \(a_1, \ldots, a_n\) stand in the relation (or, for \(n = 1\), has the property) specified by the predicate \(p\) in the ptype. We can think of this record as a “snapshot” of parts of a situation which are relevant to the classifier in question.

5.2 The meaning of “right”

We can now say what a meaning in B’s lexicon might look like before a round of the left-or-right game. We assume that B has meanings only for “left” and “right”. In our representations of meanings, we will combine the TTR representations of meanings with the TTR representation of classifier perceptrons.

Agent B’s initial take on the meaning of “right” is represented by the record in (26)\(^{11}\).

\[
(26) \quad [\text{right}]^B =
\begin{align*}
\text{w} = & \begin{bmatrix}
0.800 \\
0.010
\end{bmatrix} \\
\text{t} = & 0.090 \\
\text{bg} = & \begin{bmatrix}
\text{sr}_\text{pos} : & \text{RealVector} \\
\text{foc} : & \text{Ind} \\
\text{spkr} : & \text{Ind}
\end{bmatrix} \\
\text{r} = & \lambda \text{r} : \text{bg} \left( \begin{bmatrix}
\text{c}_\text{perc} = & \text{sr}_\text{pos} = \text{r}.\text{sr}_\text{pos} : \pi_{\text{right}}(\text{w}, \text{t})(\text{r}) \\
\text{foc} = & \text{r}.\text{foc}
\end{bmatrix} \right) \left( \begin{bmatrix}
\text{c}_\text{tell} = & \text{str} = \text{“right”} \\
\text{spkr} = & \text{r}.\text{spkr} \\
\text{foc} = & \text{r}.\text{foc}
\end{bmatrix} : \text{right}(\text{r}.\text{foc}) \right)
\end{align*}
\]

where

\[
\pi_{\text{right}} : \text{RealVector} \rightarrow \text{Real} \rightarrow \begin{bmatrix}
\text{sr}_\text{pos} : & \text{RealVector} \\
\text{foc} : & \text{Ind}
\end{bmatrix} \rightarrow \text{Type}
\]

such that

\[
\text{if } \text{r} \left( \begin{bmatrix}
\text{sr}_\text{pos} : & \text{RealVector} \\
\text{foc} : & \text{Ind}
\end{bmatrix} \right), \text{ then } \pi_{\text{right}}(\text{w}, \text{t})(\text{r}) = \begin{cases}
\text{right}(\text{r}.\text{foc}) & \text{if } \text{r}.\text{sr}_\text{pos} \cdot \text{w} > \text{t} \\
\neg \text{right}(\text{foc}) & \text{otherwise}
\end{cases}
\]

We will abbreviate this as shown in (27).

\(^{11}\)We are here assuming that we have a definition of dot product for TTR vectors \(a: \text{RealVector}_n\) and \(b: \text{RealVector}_n\) such that \(a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n\). We also implicitly assume that the weight vector and the sensor reading vector are both of type \(\text{RealVector}_n\), for some \(n\).
The fields $w$ and $t$ specify weights and a threshold for a classifier perceptron which is used to classify sensor readings. The $bg$ field represents constraints on the input context, which requires that there is a colour sensor reading and a focused object $foc$.

The $f$ field specifies the static meaning function, taking a record of the type specified by the $bg$ field to a record type containing two fields. Firstly, the value of $c_{\text{perc}}^{\text{right}}$ is a proof of either $\text{right}(foc)$ or $\neg \text{right}(foc)$, depending on the output of the classifier perceptron which makes use of $w$ and $t$. Here, $\text{right}(x)$ for some $x:\text{Ind}$ is a perceptual ptype (a type constructed from a predicate), and objects of this type are proofs that $x$ is (to the) right. As a proof of $\text{right}(foc)$ we count a snapshot of relevant parts of the situation, consisting of the current sensor reading and a specification of the currently focused object. Secondly, the value of $c_{\text{tell}}^{\text{right}}$ is a record containing information about an utterance, namely that a speaker just uttered the word “right”. We assume that this counts as a proof that foc is (to the) right. This implements an assumption that whatever A says is correct, an assumption that one could choose to remove in a more complicated version of the left-or-right game.

5.3 Contextual interpretation of “right”

We will first show a case where interpretation runs smoothly. Player A picks up an object and places it in the frame, and B finds the object and assigns it the individual marker obj$_45$, directs the position sensor to it and gets a reading. Player A now says “right”, after which B’s take on the situation is $s^B_1$, repeated here for convenience:

\[
\left[\begin{array}{l}
\text{srpos} = [0.900 0.100] : \text{RealVector} \\
\text{foc} = \text{obj}_45 : \text{Ind} \\
\text{spkr} = A : \text{Ind}
\end{array}\right]
\]

To interpret A’s utterance, after checking that $s^B_1 \sqsubseteq [\text{right}]^B bg$, B computes $s^B_1 \land [\text{right}]^B fg[bg/s^B_1]$ to yield a new take on the situation $s^B_2$:

\[
\left[\begin{array}{l}
\text{srpos} = [0.900 0.100] : \text{RealVector} \\
\text{foc} = \text{obj}_45 : \text{Ind} \\
\text{spkr} = A : \text{Ind}
\end{array}\right]
\]
Here, the classifier takes $s_1^B$ to contain a proof of right(obj45). A partial derivation of $s_2^B$ is shown in (30).

\[(30) \quad T_2^B = T_1^B \land \mathcal{F}(\text{right})^B.f) = \]
\[
\begin{align*}
&\text{sr}_\text{pos} = [0.900 \quad 0.100] : \text{RealVector} \\
&\text{foc} = \text{obj45} : \text{Ind} \\
&\text{spkr} = \text{A} : \text{Ind} \\
&c_{\text{perc}}^\text{right} = \begin{bmatrix} \text{sr}_\text{pos} = [0.900 \quad 0.100] \end{bmatrix} : \pi_{\text{right}}(0.800 \quad 0.010, 0.090)(\begin{bmatrix} \text{sr}_\text{pos} = [0.900 \quad 0.100] \\ \text{foc} = \text{obj45} \\ \text{spkr} = \text{A} \end{bmatrix}) = \\
&c_{\text{tell}}^\text{right} = \begin{bmatrix} \text{str} = \text{“right”} \\ \text{spkr} = \text{A} \\ \text{foc} = \text{obj45} \end{bmatrix} : \text{right(obj45)}
\end{align*}
\]

The interaction so far is shown in (31), with the classifier represented as a line splitting the frame into a “left” and a “right” section. (Training the classifier will result in this line being moved and/or tilted.) As can be seen, the object placed by A is clearly within the area classified as “right”, and B consequently responds “okay”.

\[(31) \quad \text{A places object:}
\]

\[
\begin{array}{c}
\begin{array}{c}
\text{+} \\
\text{•}
\end{array}
\end{array}
\]

A: “right”
B: “okay”

A’s updated take on the context, $s_2^B$, now contains two fields whose type is right(obj45), namely $c_{\text{perc}}^\text{right}$ and $c_{\text{tell}}^\text{right}$. The value of $c_{\text{perc}}^\text{right}$ is a proof of right(obj45). The record acting as a proof of right(obj45) can be thought of as a “snapshot” of relevant parts of the situation, consisting of the current sensor reading and a specification of the currently focused object. The value of $c_{\text{tell}}^\text{right}$ is a record containing information about an utterance, namely that A uttered the word “right” when foc was in focus. We assume that this counts as a proof that foc is (to the) right. This implements an assumption that A is always right, an assumption that one could choose to remove in a more complicated version of the left-or-right game.

---

13For brevity, we here use an extremely simplified representation of linguistic structure (a string and a speaker). This is a placeholder for a more complete representation such as that proposed in Cooper (2008).
6 Learning perceptual meaning from interaction

So far, we have seen how contextual utterance interpretation works when everything proceeds smoothly. In such cases, it may not be necessary to modify the meanings involved (although one could choose to reinforce existing meanings whenever they are successfully put to use). However, sometimes a DP’s take on the meaning of an expression does not quite match a situation where it is used, and in such cases it may be beneficial to the DP to learn from this.

6.1 Empirical evidence for semantic learning from interaction

To show how meaning can be learnt by observations of language use in interaction, Carey and Bartlett (1978) set up an experiment based on nonsense words to mimic the circumstances in which children naturally encounter new words. The subjects were 3- and 4-year olds. To enable testing for learning effects, they used the nonsense word: “chromium” to refer to the colour olive. They use it in the normal (nursery school) classroom activity of preparing snack time. In the experiment, one cup painted olive, and another was painted red. The adult test leader says to the child “Bring me the chromium cup; not the red one, the chromium one.”. All children picked the right cup; however, this could be done by focusing on the contrast “not the red one” without attending to the word “chromium”.

The results indicated that a very low number of exposures (five) had influenced the child’s naming of olive and had effected a lexical entry for “chromium” which in many cases included that it was a colour term, and in some cases knowledge of its referent. Some learning seems to occur after a single exposure, at least sometimes. Based on this, it was concluded that acquisition proceeds in two phases. Firstly, fast mapping resulting from one or a few exposures to the new word; this includes only a fraction of the total information constituting full learning of the word, and typically includes hyponym relations. Secondly, extended mapping over several months, by which children arrive at full acquisition, including the ability to identify and name new instances.

The experiment demonstrates learning meaning from interaction, and shows that both ontological information and perceptual aspects of meaning can be learnt in this way. In previous work (Larsson, 2010), we have sketched a formal account of fast learning of ontological meaning from interaction. Here, we will focus on learning of those aspects of meaning which are related to perception, e.g. learning to identify new instances of the colour “chromium”. This will require integrating low-level perceptual aspects of meaning into a formal semantics framework.

6.2 Detecting inconsistency in interpretation

One thing that can go wrong in interpreting an utterance is that the output context is inconsistent. This is generally caused by the foreground meaning of an utterance being inconsistent with the context in which it was spoken. We refer to this as foreground inconsistency to distinguish it from other kinds of problems potentially arising from contextual interpretation\(^{14}\). Given our formulation of the context update function corresponding to meanings, we can formally capture foreground inconsistency between a situation \(s\) and the meaning of an expression \(e\) as in (32).

(32)  Foreground inconsistency:  \(s \land \mathcal{F}(\llbracket e \rrbracket.f) \approx \bot\)

For example, take \(e\) to be “The man runs” as above in (14), and \(s\) as in (33).

\(^{14}\)The others are type mismatch and background inconsistency; since these are not immediately relevant to the left-or-right game they will be described elsewhere.
In this case, contextual interpretation would play out as in (34).

\[(34) \quad s \land F([\text{The man runs}].f) =
\]
\[
\begin{bmatrix}
\text{ref} & : & \text{Ind} \\
\text{c}_{\text{man}} & : & \text{man}(\text{ref}) \\
\text{c}_{\neg \text{run}} & : & \neg\text{run}(\text{ref})
\end{bmatrix}
\land
\begin{bmatrix}
\text{c}_{\text{run}} & : & \text{run}(s.\text{ref})
\end{bmatrix}
\]
\[
\begin{bmatrix}
\text{ref} & : & \text{Ind} \\
\text{c}_{\text{man}} & : & \text{man}(\text{ref}) \\
\text{c}_{\neg \text{run}} & : & \neg\text{run}(\text{ref}) \\
\text{c}_{\text{run}} & : & \text{run}(\text{ref})
\end{bmatrix}
\approx \bot
\]

Here, the result of contextual interpretation is a take on the context where the same man both runs and does not run, which is regarded as inconsistent. We will now see how this plays out in the left-or-right game.

### 6.3 Detecting inconsistency in the left-or-right game

We now assume that in the next round of the left-or-right-game, A places another object in a different position in the frame and again says “right”. Now, B’s take on the situation is as in Example (35).

\[(35) \quad s^B_3 =
\begin{bmatrix}
\text{sr}_{\text{pos}} = [0.100 \ 0.200] : \text{RealVector} \\
\text{foc} = \text{obj}_{46} : \text{Ind} \\
\text{spkr} = A : \text{Ind} \\
\text{perc}_{\text{right}} = \text{sensor}_{\text{pos}} = [0.900 \ 0.100] : \text{right(\text{obj}_{45})} \\
\text{tell}_{\text{right}} = \text{str} = \text{“right”} \\
\text{foc} = \text{obj}_{45} : \text{right(\text{obj}_{45})}
\end{bmatrix}
\]

Note that foc has been updated and that there is a new sensor reading¹⁵. As before, B interprets A’s utterance to yield a new take on the situation¹⁶:

---

¹⁵We are assuming that takes on situations can be updated not only by applying dynamic meanings to them, but also by applying non-monotonic updates, as in the Information State Update approach to dialogue management (Traum and Larsson, 2003). Specifically, we assume the values of sr_{pos}, foc and spkr have been updated in this way.

¹⁶We are assuming a mechanism for relabeling fields if labels conflict, by attaching an integer to the label, starting with 1.
This time, however, applying the classifier perceptron to the sensor input yields \( \neg \text{right(obj}_{46} \) and hence the classifier takes \( s^B_3 \) to contain a proof both of \( \neg \text{right(obj}_{46} \) (labelled \( c^1_{\text{perc right}} \)) and a proof of \( \text{right(obj}_{46} \) (labelled \( c^1_{\text{tell right}} \)). Basically, what is happening is that \( B \) is hearing that \( \text{obj}_{46} \) is to the right, but seeing that it is not. This is a case of foreground inconsistency – the record type \( s^B_4 \) is inconsistent (\( s^B_4 \approx \bot \)). That is, there can be no situation (record) of this type.

According to the rules of the game, \( B \) resolves this conflict by trusting \( A \)'s judgement over \( B \)'s own classification. Hence, \( B \) must remove \( c^1_{\text{perc right}} \). Furthermore, \( B \) can learn from this exchange by updating the weights used by the classifier perceptron associated with \( \llbracket \text{right} \rrbracket \).

### 6.4 Updating perceptual meaning

Perceptrons are updated using the perceptron training rule:

\[
(37) \quad w_i \leftarrow w_i + \Delta w_i
\]

where

\[
(38) \quad \Delta w_i = \eta(o_t - o)x_i
\]

where \( o_t \) is the target output, \( o \) is the actual output, \( \eta \) is the learning rate (a positive constant), and \( w_i \) is associated with input \( x_i \). Note that if \( o_t = o \), there is no learning. However, since \( B \) only tries to learn from mistakes (and not from successes), we have already established that \( o_t - o \) is 1.0 for a perceptron outputting real numbers. (In any case, our classifier perceptron instead outputs types, so it is not very useful for computing this difference.)

We can now formulate the perceptron training rule as a function taking a sensor reading \( s \) and a record specifying a weight vector \( w \) and a threshold \( t \), and and yielding a record containing updated \( w \) and \( t \) values. The training function \( \theta \) and its type is given in (39), where \( \eta \) is the learning rate and \( \eta^\theta \) is an \( n \)-dimensional real-valued vector where \( \eta^\theta_m = \eta \) for all \( m, 1 \leq m \leq n \), e.g. \( \eta^2 = [\eta \ \eta] \).

---

We here train the threshold \( t \) separately; an alternative is to include it as \( w_0 \) and assume a dummy input \( x_0 = 1 \). In the latter case, \( t \) is updated as a part of updating \( w \).
\[
\theta = \lambda s: \text{RealVector}(r; \begin{array}{l}
w : \text{RealVector} \\
t : \text{Real}
\end{array}) \rightarrow \begin{array}{l}
w = r.w + \eta^s.s \\
t = r.t - \eta
\end{array}
\]

\[
\theta: \text{RealVector} \rightarrow \begin{array}{l}
w : \text{RealVector} \\
t : \text{Real}
\end{array} \rightarrow \begin{array}{l}
w = \text{RealVector} \\
t = \text{Real}
\end{array}
\]

The output of the training function can then replace the weight vector and the threshold in the previous perceptron to yield an updated perceptron. Since the perceptron is part of the meaning of an expression, this meaning is also updated. Given a meaning \( e \) containing a perceptron \( p \), and a sensor reading \( s \), the updated meaning \( e' \) is computed as in (40).

\[
e' = e \wedge \theta(s, \begin{array}{l}
w = e.w \\
t = e.t
\end{array})
\]

We here use the asymmetric merge operator \( \wedge \) which is similar to the merge operator \( \wedge \) but gives precedence to the argument to the right of the operator in cases where both arguments specify a field with the same label\(^{18}\). In the example above we have \( e = \text{right} \), and \( s = s^B_{\text{pos}} \). For \( \eta = 0.1 \) we get

\[
\text{right}^B = \text{right}^B \wedge \theta(s^B_{\text{pos}}, \begin{array}{l}
w = \text{right}^B.w \\
t = \text{right}^B.t
\end{array})
\]

B thus updates the meaning of “right” by modifying the weight vector used by a classifier perceptron, based on the output of applying the dynamic semantics of “right” to B’s take on the situation. Example (42) shows agent B’s revised on the meaning of “right”, \( \text{right}^B \):  

\[
\text{right}^B = \begin{array}{l}
w = 0.808 \\
t = -0.010
\end{array}
\]

\[
\text{bg} = \begin{array}{l}
\text{sr}_{\text{pos}} : \text{RealVector} \\
\text{foc} : \text{Ind} \\
\text{spkr} : \text{Ind}
\end{array}
\]

\[
\text{perc}_{\text{right}} = \begin{array}{l}
\text{sr}_{\text{pos}} = \text{bg} \cdot \text{sr}_{\text{pos}} \\
\text{foc} = \text{bg} \cdot \text{foc}
\end{array} ; \begin{array}{l}
\text{right}(\text{r} \cdot \text{foc}) \quad \text{if} \quad \text{r} \cdot \text{sr}_{\text{pos}} \cdot w > t \\
\neg \text{right}(\text{r} \cdot \text{foc}) \quad \text{otherwise}
\end{array}
\]

\[
f = \lambda r: \text{bg}(\begin{array}{l}
\text{str} = \text{“right”} \\
\text{spkr} = \text{bg} \cdot \text{spkr} \\
\text{foc} = \text{bg} \cdot \text{foc}
\end{array}) : \text{right}(\text{bg} \cdot \text{foc})
\]

In (43), the interaction including both the first and the second round is shown. The second object placed by A is originally classified as being to the left of the line representing the classifier for “right”, which triggers B to retrain the classifier; as a result, the second object ends up being on the right side of the line representing the classifier\(^{19}\). Correspondingly, B utters “aha” rather than “okay” as in the first round.

\(^{18}\)More precisely, if \( r_1 \) and \( r_2 \) are records, \( r_1 \wedge r_2 \) denotes the asymmetric merge of \( r_1 \) and \( r_2 \). If \( r_1 \) and \( r_2 \) are both records, then for any label \( \ell \) which occurs in both \( r_1 \) and \( r_2 \), \( r_1 \wedge r_2 \) will contain a field labelled \( \ell \) with the type resulting from the asymmetric merge of the corresponding values in the \( \ell \)-fields of the two records (in order). For labels which do not occur in both records, \( r_1 \wedge r_2 \) will contain the fields from \( r_1 \) and \( r_2 \) unchanged. If one or both of \( r_1 \) and \( r_2 \) are non-records, then \( r_1 \wedge r_2 \) will be \( r_2 \). Our notion of asymmetric merge is related to the notion of priority unification (Shieber, 1986). Cooper (in progress) also defines asymmetric merge for record types; however, this is not needed for our current purposes.

\(^{19}\)In general, however, there is no guarantee that retraining the classifier will yield a result consistent with A’s utterance. In cases where inconsistency remains after retraining, B could simply discard this perceptual classification as faulty.
7 From learning to coordination

In the left-or-right game, as described above, there is a fundamental asymmetry in that agent A is assumed to be fully competent at judging whether objects are to the right or not, whereas agent B is to learn this. By contrast, when humans interact they mutually adapt to each others’ language use on multiple levels; this is known variously as alignment (Pickering and Garrod, 2004), entrainment (Brennan, 1996) or coordination (Garrod and Anderson, 1987; Larsson, 2007). We have previously described various aspects of semantic coordination in the TTR framework (Larsson and Cooper, 2009; Cooper and Larsson, 2009; Larsson, 2009, 2010, 2011b). In Fernández et al. (2011), semantic coordination is described as a case of as reciprocal learning, i.e., a process where two (or more) agents incrementally learn from each other. In such a setting, the end product (in our case, the meanings of the expressions used by the agents) emerges from interaction, and may not exist before the interaction or be predictable from what exists before the interaction, i.e., the meanings ascribed to expressions by the individual agents. This is different from the left-or-right game as described above, where the meaning of “right” which will be shared by A and B after a sufficient number of rounds will be identical to the meaning ascribed initially by A.
7.1 Semantic coordination in dialogue

As noted above and in Larsson (2010), DPs can learn meanings by observing language addressed to them (optionally responding by uttering the word which was used in a novel way). However, several other mechanisms are available for the learning of meanings from interaction dialogue. These include corrective feedback, where one DP implicitly corrects the way an expression is used by another DP (Father’s first utterance in the dialogue below, taken from Clark (2007)), as well as explicit definitions of meanings (Father’s second utterance).

(44) “Gloves” example:

Naomi: mittens
Father: gloves.
Naomi: gloves.
Father: when they have fingers in them they are called gloves and when the fingers are all put together they are called mittens.

Here are a few more examples of corrective feedback in first language acquisition:

(45) A: That’s a nice bear.
B: Yes, it’s a nice panda.
Abe: I’m trying to tip this over, can you tip it over? Can you tip it over?
Mother: Okay I’ll turn it over for you.
Adam: Mommy, where my plate?
Mother: You mean your saucer?
Naomi: Birdie birdie.
Mother: Not a birdie, a seal.

The first one is made up, the others are quoted from various sources in Clark and Wong (2002) and Clark (2007). In general, corrective feedback can be regarded as offering an alternative form to the one that the speaker used. In Larsson and Cooper (2009), we sketch a formal account of learning of various aspects of word meaning from corrective feedback and explicit definition.

We regard the learning of meanings in first language acquisition as a special case of semantic coordination, where there is a clear asymmetry between the agents involved with respect to expertise in the language being acquired when a child and an adult interact. However, we believe that the mechanisms for semantic coordination used in these situations are similar to those which are used when competent adult language users coordinate their language. For example, Jefferson (1987) provides the following example of what she refers to as “embedded correction” (roughly equivalent to what we here call corrective feedback) in from a conversation in a shop²⁰.

---
²⁰ As we can note from this example, semantic coordination involves coordination of mappings from words to meanings as well as coordination of the meanings themselves. Note, however, that changing the mapping from words to meanings (as when coordinating on “threads” as more appropriate than “wales”) also involves changes to the meanings involved, provided that meanings include conditions on contexts of appropriate use.
(46) Customer: Mm, the wales are wider apart than that.
Salesman: Okay, let me see if I can find one with wider threads
((looks through stock))
Salesman: How’s this.
Customer: Nope, the threads are even wider than that.

The kind of learning we see in first language acquisition is not limited to what we usually think of as acquisition, but occurs in everyday conversation to a greater or lesser degree. We are constantly in the process of coordinating the meanings of the linguistic expression that we use in communication. Two agents do not need to share exactly the same linguistic resources (grammar, lexicon etc.) in order to be able to communicate, and an agent’s linguistic resources can change during the course of a dialogue when she is confronted with a (for her) innovative use.

7.2 Semantic coordination and the nature of linguistic meaning

One may choose to apply the theory outlined here only to cases of language learning, or one could take them to be significant of language in general. We believe that accounting for learning of meanings in formal semantics is fundamental to understanding the nature of linguistic meaning. We regard the meaning of a linguistic expression or word to depend on previous uses of that word. Hence, meaning is inherently plastic and learning is central to the notion of meaning. We take the primary source of learning to be linguistic interactions involving an agent as a participant (although passively observed language may also play a role). This view implies that a central task of semantic theory is to model semantic plasticity and semantic coordination, i.e. how meanings change as a result of language use in interaction.

As seen above, we assume that speakers have internalised representations guiding their use and interpretation of specific words. These representations depend, among other things, on observations of previous situations where the word in question has been used, and on generalisations over these situations. Semantic coordination is described in terms of updates to individual meaning representations associated with words triggered by observations of their use in dialogue.

There are many mechanisms for semantic coordination, including clarification requests, explicit corrections, meaning accommodation, and explicit negotiation. Semantic coordination, in turn, is a kind of language coordination (other kinds include e.g. phonetic coordination). Finally, language coordination coexists with information coordination, the exchanging and sharing of information (agreeing on relevant information and future action; maintaining a shared view on current topics of discussion, relevant questions etc.).

Semantic coordination happens in dialogue; it is part of language coordination; and it is a prerequisite for information coordination. If we say that a linguistic expression $e$ has meaning only if it is possible to exchange information using $e$, then semantic coordination is essential to meaning. A linguistic expression $e$ has meaning in a language community when the community members are sufficiently coordinated with respect to the meaning of $e$ to allow them to use $e$ to exchange information. In other words: meaning emerges from a process of semantic coordination in dialogue.

By modelling how individuals (1) represent and (2) coordinate on meanings, we indirectly model the emergence, perpetuation and variation of meaning in a linguistic community. Although our representations

---

21This idea is not new. Its origins can be traced back to the idea that “meaning is use” (Wittgenstein, 1953).
22We use this term instead of “dynamic” to avoid confusion with “dynamic semantics”.
23In fact “dispositions” may be a more accurate word. What matters for semantic coordination is not the meaning representation as such, but rather how the agent uses and interprets the expression in question. In the end, given that we do not have direct access to each other’s (or indeed our own) meaning representations, all that can be coordinated on is observable behaviour (including utterances attempting to provide explicit definitions, etc.) This means that two agents may entertain radically different meaning representations but still use and interpret utterances in a coordinated way. An example of this is successful cases of dialogue systems interacting with humans using spoken natural (or at least approximately natural) language.
concern individual agents, meaning itself is inherently social and dependent on learning and adaptation through interaction.

8 Compositionality for perceptual meaning

Keeping within a very simple language game, we have shown how perceptual and symbolic meanings can be combined in a single framework. We believe that the account of perceptual meaning presented above can be useful in connecting low-level (“subsymbolic”) aspects of meaning explicitly to formal semantics in a detailed and integrated manner. Doing this would ideally enable making use of the rich body of work in formal semantics of natural language from the last 50 or so years in the development of a more complete account of meaning which would combine both high-level (logical-inferential) and low-level (perceptual) aspects. A crucial next step is to demonstrate how the principle of compositionality, which is at the heart of formal semantics, can be applied also to subsymbolic aspects of meaning.

8.1 Related work

As has already been mentioned, perceptual aspects of meanings have been explored in previous research (Barsalou et al., 2003; Roy, 2005; Steels and Belpaeme, 2005; Kelleher et al., 2005; Škočaj et al., 2010). However, the connection to logical-inferential meaning and compositionality as traditionally studied in formal semantics has not been a focus of this body of work. There have also been attempts to extend semantic formalisms to cover embodied meaning (Feldman, 2010), but this line of work has tended to concentrate on abstract (high-level) representations and has generally not paid not attention to low-level perceptual aspects of context.

In computational linguistics, a well-known approach to dealing with low-level meaning aspects is the Vector Space Model (Turney and Pantel, 2010), where the context of linguistic expressions are encoded as vectors in a space, whose dimensions typically represent co-occurring linguistic expressions. In principle such a model could also represent non-linguistic aspects of the context such as perceptually salient objects and relations. VSMs can also represent semantic gradience phenomena, and offer an account of learnability of meanings (van Eijck and Lappin, 2012). There has also been work on compositionality for VSMs (Mitchell and Lapata, 2008).

Although compositional VSMs can in principle represent non-linguistic aspects of context, it is not clear how to the vectors resulting from compositional analysis are to be interpreted (van Eijck and Lappin, 2012). Furthermore, it is not clear how VSMs could be put to work in something like the left-or-right game. We do not exclude the possibility that VSMs representing perceptually available aspects of context could be used as a component of a classifier for perceptual input. In fact, on a wide enough notion of VSM perhaps even the perceptron classifier described above could be considered a VSM; after all, it does represent an aspects of context (position) as a vector in a 2-dimensional space.

However, the standard approach to compositionality in the VSM framework is to compose vectors together in various ways. As mentioned, it is not so clear how the resulting vectors are to be interpreted - for example, what role they play in inferences of the kind typically studied in formal semantics. The key to dealing with compositionality for perceptual meaning, we believe, is to do compositionality not on the low-level vectors, but on the level of classifiers. We thus take it that the perceptual meaning of a sentence can be composed from the perceptual meanings of the constituent expressions of the sentence.

8.2 Compositionality in the left-or-right game

Exploring compositionality in something like the left-or-right game requires extending it. An obvious extension is to add more words (e.g. “upper” and “lower”) and some simple grammar (“upper left”, “lower
right” etc). Furthermore, additional sensors and classifiers, e.g. for colour, shape and relative position, can be added, thus enabling meanings of colour and shape terms as well as complex phrases like “the green box is to the left of the upper red circle”. The next step would be to add motor aspects of meaning to capture meanings of action verbs such as “throw”. One could imagine gradually extending the scope of this game towards something resembling human language.

As a simple proof of concept of compositionality in the framework put forward above, we will show how to compute the meaning of “upper right” from the meanings of “upper” and “right”. The meaning “upper” can be represented thus (ignoring the exact details of the classifier):

\[
\text{w}_{\text{upper}} = \ldots
\]
\[
\text{t}_{\text{upper}} = \ldots
\]
\[
\text{bg} = \begin{bmatrix}
\text{sr}_{\text{pos}} & \text{RealVector} \\
\text{foc} & \text{Ind} \\
\text{spkr} & \text{Ind}
\end{bmatrix}
\]
\[
\text{f} = \lambda r : \text{bg}(\begin{bmatrix}
\text{w}_{\text{upper}} = \ldots \\
\text{t}_{\text{upper}} = \ldots \\
\text{w}_{\text{right}} = \ldots \\
\text{t}_{\text{right}} = \ldots \\
\text{sr}_{\text{pos}} & \text{RealVector} \\
\text{foc} & \text{Ind} \\
\text{spkr} & \text{Ind}
\end{bmatrix}
\]
\[
\text{f} = \lambda r : \text{bg}(\begin{bmatrix}
\text{str} = \text{“upper”} \\
\text{spkr} = r.\text{spkr} \\
\text{foc} = r.\text{foc}
\end{bmatrix}
\]
\[
\text{f} = \lambda r : \text{bg}(\begin{bmatrix}
\text{str} = \text{“right”} \\
\text{spkr} = r.\text{spkr} \\
\text{foc} = r.\text{foc}
\end{bmatrix}
\]

Given this, the compositional meaning of “upper right” is simply computed by merging the meanings of “upper” and “right” as in (48)

\[
\text{w}_{\text{upper}} = \ldots
\]
\[
\text{t}_{\text{upper}} = \ldots
\]
\[
\text{w}_{\text{right}} = \ldots
\]
\[
\text{t}_{\text{right}} = \ldots
\]
\[
\text{bg} = \begin{bmatrix}
\text{sr}_{\text{pos}} & \text{RealVector} \\
\text{foc} & \text{Ind} \\
\text{spkr} & \text{Ind}
\end{bmatrix}
\]
\[
\text{f} = \lambda r : \text{bg}(\begin{bmatrix}
\text{w}_{\text{upper}} = \ldots \\
\text{t}_{\text{upper}} = \ldots \\
\text{w}_{\text{right}} = \ldots \\
\text{t}_{\text{right}} = \ldots \\
\text{sr}_{\text{pos}} & \text{RealVector} \\
\text{foc} & \text{Ind} \\
\text{spkr} & \text{Ind}
\end{bmatrix}
\]
\[
\text{f} = \lambda r : \text{bg}(\begin{bmatrix}
\text{str} = \text{“upper”} \\
\text{spkr} = r.\text{spkr} \\
\text{foc} = r.\text{foc}
\end{bmatrix}
\]
\[
\text{f} = \lambda r : \text{bg}(\begin{bmatrix}
\text{str} = \text{“right”} \\
\text{spkr} = r.\text{spkr} \\
\text{foc} = r.\text{foc}
\end{bmatrix}
\]

(Note that we are now indexing weights and thresholds for different classifiers.)

We are here assuming that “upper right” means, simply “upper and right”, or more explicitly “(in the)

\footnote{When merging two types $T_1$ and $T_2$, the merge operation is applied recursively to any fields occurring in both $T_1$ and $T_2$ (see Cooper (2005)). Merge for functions is defined as $\lambda r : T_1 (T_2) \land \lambda r : T_3 (T_4) = \lambda r : T_1 \land T_3 \land T_2 \land T_4$.}
upper (part of the frame) and (to the) right. For simplicity, we have here ignored matters of word order so that “upper right” and “right upper” are both taken to have the same meaning; the constraint $c_{\text{tell upper}}$ only states that the string “upper” was uttered and $c_{\text{upper}}$ that “right” was uttered. Adding a grammar component would prevent syntactically anomalous expressions.

A further simplification is that we assumed that both expressions (“upper” and “right”) referred to the same object, the object in the shared focus of attention. In cases where more than one object is involved, e.g. in relations such as “$x$ is to the right of $y$”, things will get a bit more complex since the origin of each sensor reading ($x$ or $y$) must now be explicitly represented.

The resulting representation will yield a plausible interpretation in cases where the focused object is classified as being both to the right and in the upper part of the frame (given the intuitive setup of the “upper” classifier). Note that instead of trying to somehow compute compositionality on the level of vector representations, we instead computed it on the higher level of classifiers. Nevertheless, what we get is a semantic representation that will indeed classify situations where the focused object is to the upper left, based on real-valued vectors. Although the same sensor reading was used for both classifiers in the example, the approach straightforwardly extends to compositional expressions involving several different sensors.

### 8.3 Compositionality for degree modifiers and context sensitive meanings

Of course we are not saying that all compositionality will be as trivial as in this example. For example, to correctly deal with a degree modifier such as “far right” one let the meaning of “far” modifying some parameter of the “right” classifier, rather than simply adding a further condition. To illustrate this idea, we assume that the compositional meaning of “far right” is computed as shown in (49).

$$\text{(49) } \llbracket \text{far right} \rrbracket = \llbracket \text{far} \rrbracket . f(\llbracket \text{right} \rrbracket)$$

This assumes that the meaning of “far” specifies a function $f$ which is to be applied to the meaning of “right” (or whatever word appears adjacent to “far”). The function $f$ in (50) takes a record $m$ (specifying the meaning of the adjacent word) containing a classifier $\pi$ with a threshold $t$ of type Real, and yields a new meaning identical to that of the adjacent word except that the classifier threshold has been replaced by the product of $t$ and a factor $\alpha$ ($\alpha \geq 1$).

$$\text{(50) } \llbracket \text{far} \rrbracket = \begin{cases} \alpha = 1.4 \\ f = \lambda m: \begin{cases} t : \text{Real} \\ (m \sqcap t = \alpha \cdot m.t) \end{cases} \end{cases}$$

This would give a meaning of “far right” (given our original meaning of “right”) as in (51). This corresponding schematically to a classifier where the line dividing “right” from “not right” has been moved to the right (compared to the classifier for “right”), as shown in (52).

---

25. This may or may not correspond to the use of “upper right” in ordinary everyday English. It is quite possible that the everyday meaning of “upper right” is not identical to the conjunction of “upper” and “right”, but instead depends in a more complicated way on the meanings of “upper” and “right”.

26. This requires a slight modification of the meaning of “right” where the threshold is parametrized:
Also, there are the well-known cases of conceptual blending (Fauconnier and Turner, 2008), for example that “red” in “red wine” does not mean the same as “red” in “red hair”. Here, the ability to mix high-level and low-level information in TTR comes in handy. The classifier connected to “red” could be modified, or altered, depending on the presence of high-level information concerning the kind of entity to which redness was ascribed. A sketch of how this could be done in TTR is shown in (53)\(^\text{27}\) (where \(p_1, \ldots, p_n\) are parameters used for colour classification).

In (53), the bg field contains (in addition to the focused object and a colour sensor reading in the form of a real-valued vector) a field \(c_{\text{kind}}\) specifying the kind of entity that foc is. In the fg field, this information is used to determine which classifier of redness to use, and if no relevant kind has been specified, a default “red” classifier is used. The relation between the redness classifiers is independent of how they are used; they could either be separately trained or related using some transformation function mapping the ordinary redness classifier onto a subregion of the colour space.

\(^{27}\)There are of course also other possible analyses of “red”, for example that the word is ambiguous. Here we are assuming that the word is not ambiguous but that its meaning is context-dependent.
9 Requirements on formal theories accounting for learning of perceptual meaning

As we argue in Larsson (2011a), the view of intensional meaning as involving sensors and classifiers interfacing with perception has important consequences for how we think about extensions. In classical model-theoretic semantics, the extension of an expression, e.g. “dog” is the set of objects (dogs) in the world falling under the expression, and this set is given by a (static) function. The status of extensions in the present theory, at least in cases where the intension of an expression involves a classifier of perceptual input, is quite different. First of all, classifiers are implemented in individual agents, and in a population of agents there may be some variation in how objects are classified. Second, the only way to find out if an individual is in the intension of an expression is to apply the classifier to a situation involving the individual. Classification is an event which takes occurs in real time, which means that computing the extension of an expression is a process which takes time and not just an abstract mathematical operation. Third, classifiers may change as a result of observations of the world and interaction with other agents. Computing the universal extension of e.g. “dog”, i.e. the set of all dogs in the world, requires applying something like a “dog-classifier” to all individuals in the world that may conceivably be considered dogs. To make matters even worse, during this potentially quite lengthy process the dog-classifier may be altered, which may render previous classifications invalid.

Finally, real-world perceptual systems involving sensors and perceptual classifiers are always to some degree unpredictable, since small differences in the perception process may have large effects on its output. For example, in the left-or-right game, the classification of an object placed very near the border between “right” and “not right” may go either way depending on e.g. light conditions affecting B’s visual sensor. In general, many kinds of variations in the perceptual context can affect classification. Such variations can be regarded as a kind of noise, and classifiers are always more or less sensitive to this kind of noise. It is well known that in humans, high-level semantic knowledge can affect low-level perception of e.g. colours (Stroop, 1935; McClelland and Rogers, 2003; Geisler and Diehl, 2003; Bar, 2004; Kubat et al., 2009). This is not a design flaw so much as an important feature, as it increases the robustness of the perceptual system in noisy and dynamically changing contexts. High-level context effects may compensate for some low-level context effects but does not eliminate them, and what we end up with is are perceptual classifiers (and consequently, extensions) which are sensitive to a wide variety of high-level and low-level features of the context.

One may try to salvage a notion of definite extension corresponding to an intension by relativising it to a certain agent in a certain context at time and place. The extension of a linguistic expression would then be all the entities (objects, situations) that would be classified by that agent as falling under the expression in question, if the entity were to be presented to the agent in that specific context at that specific time and place. Note, however, that this requires a counterfactual reasoning involving modifying the context, thus potentially affecting the results of classification. It seems to use that given the above, the notion of an extension of an expression can only be regarded a theoretical idealisation whose import for language use in human interaction and hence natural language semantics is, at best, unclear.

Accounting for the phenomena of semantic coordination, semantic plasticity and perceptual meaning pose certain requirements on any formal theory purporting to account for them. Accounting for learning in individual agents requires relativising interpretation processes to individual agents. Furthermore, we believe that intensions of linguistic expressions are best treated as first-class objects, and that these types need to be structured. The reason is that accounting for semantic learning in an intuitively plausible manner requires the possibility of modifying intensions, and only structured objects can be modified. Finally, as some meanings involve classifying situations in the world based on perceptions thereof, and since classification of real-world situations is (in general) a complex and context-sensitive process involving perception of real-world situations, any formal theory of semantic plasticity should allow for some degree indeterminacy of extensions of linguistic expressions.
An influential semantic theory for discourse and dialogue, SDRT (Asher and Lascarides, 2003), offers a language for representing the logical form of discourse and of dialogue, and assigns this language a dynamic semantic interpretation – the SDRS (Segmented Discourse Representation Structure) resulting from parsing and integrating all the utterances in a dialogue. While relying on classical extensional model-theoretic semantics, SDRT has some features which may be useful if one would extend it to account for semantic plasticity and perceptual meaning. It insists on the utility of a level of representation between language and model, namely the language of SDRSs. Also, SDRSs are structured meaning representations (although encompassing only logical-inferential aspects of meaning). This opens for the possibility to recast SDRT in a type-theoretic framework, thereby making it better equipped to deal with semantic plasticity. Recent work on a type-theoretic account of word meaning by Asher (2010) is encouraging in this respect.

While the question of whether tradition model theoretic semantics can be revised to account for perceptual meaning, semantic plasticity and semantic coordination is an interesting research problem in its own right (and one that we hope to address in future work), we believe there are also good reasons to explore alternative semantic frameworks which are designed from the outset to enable explicit representation of intensions and modelling of perception, learning and coordination. In this context, it is important to point out that it is an explicit ambition in work on TTR to keep around the accumulated insights from formal semantics in the classical model-theoretic tradition.

Incidentally, the notion of type in TTR is in (at least) one respect very different from the notion of type in traditional mathematical type theory. In the latter, the objects of any type are always assumed to be known, and traditional mathematical type theory is in this respect similar to classical model-theoretic semantics. However, type theory in general is not as dependent as model theory on having determinate extensions of expressions. Since type theory in general treats types as first-class objects, it is open to a notion of types whose extension is not given. To replace the notion of extension as static, absolute and determinate, one may posit a relativised notion of extension, where we talk about the extension of an expression for an agent at a certain time, but now including only the set of individuals so far classified by the agent as falling under the expression (rather than all individuals in the world which would be thus classified). Still, even this notion is slightly problematic insofar as modifications of the classifier resulting from learning may invalidate previous classification results.

10 Generalising classification

In the above, we have shown how a simple perceptron (formalised in TTR) can be used to represent subsymbolic meaning in a formal semantics framework. In our example, the perceptron associated with a linguistic expression is included in the lexical meaning for that expression. However, we of course cannot assume that the TTR perceptron as it is provides a general method for representing perceptual meaning.

As mentioned above, one possible extension of the left-or-right game would include the use of colour terms. In Steels and Belpaeme (2005), competing adaptive networks of locally reactive units (a kind of artificial neuron) are used for classifying data from a robot’s visual perception of colour samples. Each colour term is associated with an adaptive network outputting a continuous value (rather than 0 or 1 as in the perceptron). When classifying a colour sample, the colour term associated with the network with the highest output wins over competing colour terms. A consequence of such a setup is that the classification of perceptual data as an instance of a single linguistic expression (e.g. “red”) of the relevant category (e.g. colour terms) involves all the classifiers associated with terms of the relevant category. For example, to classify a colour sample as “red”, the neurons associated with “green”, “blue” and so forth must all be fed the incoming perceptual data to see which neuron gives the highest output. That is, there is a single (complex) classifier for colour terms, which includes as parts classifiers related to different colour terms. In cases such as this, it is less clear that it is a good idea to include classifiers directly in the meaning representations of
linguistic expressions. Instead, it may be useful to represent classifiers separately from meanings, and to link representations of linguistic expressions to these classifiers.

Furthermore, perceptrons are just one kind of classifier (and a very limited one at that). We believe that the account presented here can be straightforwardly generalised to work with any kind of classifier whether based on neural networks, Bayesian reasoning, decision trees, vector spaces, etc.. These can either be represented wholly in TTR (which may or may not be feasible and/or desirable) or plugged into the TTR framework as an external resource. The only thing which is really required is that the classifier is a well-defined computable function which takes low-level (e.g. real-valued) input and produces something which can be converted into a TTR ptype as output. Furthermore, if we are interested in modelling learning, it should be possible to update the classifier based on experience.

11 Summary

The work presented here is part of a research agenda aiming towards a formal account of multi-dimensional meanings, including perceptual aspects of meaning, and semantic coordination in dialogue. In this paper, we have shown how a simple classifier of low-level real-valued information based on the Perceptron can be cast in TTR (Type Theory with Records) and thus interfaced with high-level interpretation and reasoning (involving quantification, modality, etc.). The perceptual meaning of an expression is represented by a classifier which is used to determine whether the expression holds of (or more generally, can be appropriately used in) some situation. To put this in slogan form, perceptual meanings are classifiers of sensory data. High-level (“symbolic”) and low-level (“subsymbolic”) meanings are represented in a single framework (TTR), providing a solid connection between work in formal semantics and work on perceptual aspects of meaning. Meanings are coordinated in interaction between agents; each agent has their own take on the meaning of an expression and insofar as several agents coordinate their takes on the meaning of an expression, this expression will acquire meaning in that group of agents. Both high-level and low-level aspects of meaning are regarded as dynamic entities (semantic plasticity) which can be learned as a result of observing language use in interaction, thereby enabling fine-grained semantic coordination. We have also sketched an account of compositionality for perceptual meanings.

Acknowledgments

This research was supported by The Swedish Bank Tercentenary Foundation Project P2007/0717, Semantic Coordination in Dialogue; VR project 2009-1569, Semantic analysis of interaction and coordination in dialogue (SAICD)M and the Centre for Language Technology (CLT) at the University of Gothenburg. Thanks to Robin Cooper, Simon Dobnik and Liz Coppock for comments on draft versions of this paper and enlightening discussions. Also thanks to anonymous reviewers of previous versions of this paper.

References


Cooper, R. (in progress). *Type theory and language - From perception to linguistic communication*.


28


Shieber, S. (1986). An Introduction to Unification-Based Approaches to Grammar. CSLI.


