Context-sensitive utterance planning for CCG

Geert-Jan M. Kruijff
Language Technology Lab
German Research Center for Artificial Intelligence (DFKI GmbH)
Saarbrücken, Germany
⟨gj@dfki.de⟩*

Abstract

The paper presents an approach to utterance planning, which can dynamically use context information about the environment in which a dialogue is situated. The approach is functional in nature, using systemic networks to specify its planning grammar. The planner takes a description of a communicative goal as input, and produces one or more logical forms that can express that goal in a contextually appropriate way. Both the goal and the resulting logical forms are expressed in a single formalism as ontologically rich, relational structures. To realize the logical forms, OpenCCG is used. The paper focuses primarily on the implementation, but also discusses how the planning grammar can be based on the grammar used in OpenCCG, and trained on (parseable) data.

1 Introduction

Conversational robots often need to carry out a dialogue with other agents while being situated in a dynamic environment. This poses an interesting challenge: For the robot to converse in a natural manner with other interlocutors its communication needs to be contextually appropriate, but referential contexts may naturally change in such a setting. The robot thus must be actively aware of the environment, and use this awareness when producing utterances.

Here, we present an approach to utterance planning where we can use context information to dynamically guide decisions we need to make during planning. These decisions are paradigmatic in nature, and get us from a logical form stating a communicative intention, to a logical form (or a set thereof) expressing the intention in a contextually appropriate way.

We specify a planning grammar as a systemic network, in the tradition of generation systems for systemic functional grammar [Mathiessen, 1983; Bateman, 1997]. We process using an agenda/chart-based algorithm, (meaning there is no "determinicity" assumption). The utterance planner itself is embedded in a distributed architecture that makes it possible to access the various models of the situated environment that the robot maintains. This way we can dynamically use contextual information during the planning process. The logical forms we operate on are all specified in a single formalism, namely Hybrid Logic Dependency Semantics (HLDS), meaning we have a representational continuum between discourse-level and utterance-level representations [Kruijff, 2001; Baldridge and Kruijff, 2002], and can guide utterance planning through content decisions made at higher levels. The logical form we obtain from the utterance planner serves as input to a separate OpenCCG realizer [White and Baldridge, 2003; White, 2004].

The resulting approach is related to [Stone and Doran, 1997; Cassell *et al.*, 2000]. We adopt their idea of an utterance as a description, generated from a communicative goal, and also use an "ontologically promiscuous" formalism for representing meaning [Hobbs, 1985]. We differ in that we separate out the realizer, though minimize the need for backtracking in the planner by allowing for multiple, alternative logical forms to be sent to the realizer, cf. [Foster and White, 2004]. Also, to establish contextual status of an entity, we can in principle use any type of model that the robot maintains of the environment, as long as we have an ontology on which we can establish a common ground in interpretation.

Our approach places us squarely in the full generation camp, but there is a continuum: Using the approach to including canned text as proposed in [Foster and White, 2004], we can freely position the actual planner between full generation and pre-baked generation. We can use the flexibility of full generation where necessary, notably to achieve contextual appropriateness, but if desired we can use more direct methods to specify content. Our approach owes its perspective to systemic approaches, particularly KPML [Bateman, 1997]. Where we differ is in the creation of, and relation between, the resources we use in the parser, the realizer, and the utterance planner: We use one and the same grammar for both parsing and realization (though with different algorithms), and we can derive the systemic network for utterance planning from this grammar (§4) to ensure that we have a single formulation of the robot's linguistic knowledge, in the form of a CCG grammar. We also point out (§4), how we can in principle train the planner, like e.g. [Stent *et al.*, 2004].

Overview In §2 we briefly discuss HLDS, and the overall architecture in which we employ the utterance planner. §3 presents the planner, focusing on the basic structure of the

*This research is supported by the EU FP6 IST IP "Cognitive Systems for Cognitive Assistants" (CoSy), FP6-004250-IP

planning grammar, and context sensitivity. §4 discusses how we can base the planning grammar on the specification of the grammar we employ for parsing and realization, and how we can in principle train the planning grammar given a corpus of (analyzable) utterances.

2 Background

2.1 Hybrid Logic Dependency Semantics

Hybrid Logic Dependency Semantics (HLDS; [Kruijff, 2001; Baldridge and Kruijff, 2002]) is an "ontologically promiscuous" [Hobbs, 1985] framework for representing the propositional content (or meaning) of an expression as an *ontologically richly sorted, relational structure*. The relational structure connects different bits of meaning using (directed) labelled edges. The labels on these edges indicate how the meaning of the *dependent* contributes to the meaning of the whole, rooted at the *head* node that governs the dependent.

This view on the representation of meaning can be traced back to various theories of valency in dependency grammar and related work on theta-frames. Under this view we obtain relatively flat representations. These flat representations are nowadays used in various grammar frameworks, and are closely related to the conceptual structures found in AI knowledge representations. [Baldridge and Kruijff, 2002; White and Baldridge, 2003] show how HLDS representations can be built compositionally with CCG using unification, and compare HLDS to other semantic formalisms like Minimal Recursion Semantics [Copestake *et al.*, 1997].

Formally, we represent the meaning of an expression using hybrid logic [Blackburn, 2000; Areces, 2000]. Being a type of modal logic, hybrid logic is ideally suited to capture relational structures. Furthermore, it adopts an approach to sorting that enables us to represent meaning as an ontologically richly sorted structure. This works out as follows.

Hybrid logic is a modal logic, but a modal logic with a twist. Modal logics are interpreted on models consisting of states and accessibility relations between these states. However, we cannot reference these states directly in the language of a modal logic itself. This is problematic. Modal logic is often used to model temporal structure, e.g. using Prior's Past and Future operators. Unfortunately, all we can express is that something happened at some point in the past, or will happen at some point in the future. We cannot *specify* that point in a formula, which is counter-intuitive; cf. [Blackburn, 2000].

Hybrid logic addresses this issue by introducing *nominals* into the language. A nominal is a type of formula, which is interpreted as a unique reference to a state in the underlying model theory of the logic. Nominals are formulas, and hence equal citizens in the language next to e.g. propositions. There are several operators that range over nominals, the most ubiquitous for our current purposes being the "@" operator: $@_n\phi$ means that "at the state referred to by n, formula ϕ holds".

We use the standard modal operators to model relations: $@_n\langle R\rangle m$ means that there is a relation R between the nominals n and m. Particularly important for our purposes is that we can sort the nominals further, to indicate the ontological sort or category of the proposition that holds at the state referred to by the nominal. For example, $@_{\{k:\mathsf{person}\}}$**Kathy** represents the fact that Kathy is a person; note that we can thus use nominals as (neo-Davidsonian style) discourse referents.

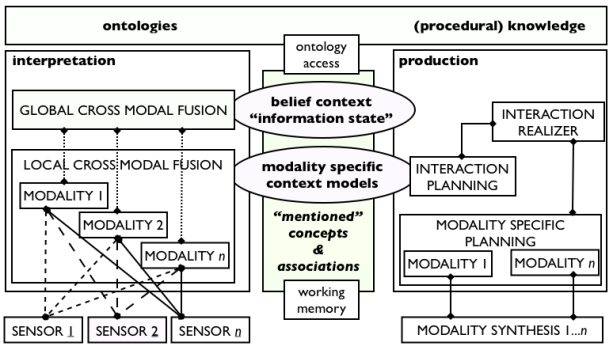

Figure 1: Conceptual architecture

We obtain a relational structure by relating nominals through modal relations, e.g. (1).

(1) a. Kathy saw Eli.

b. $@_{\{s:\mathsf{observing}\}}$(**see** $\&$ $\langle Tense\rangle$**past** $\&$ $\langle Actor\rangle(k : person \,\&$ **Kathy**$)$ $\&$ $\langle Patient\rangle(e : person \,\&$ **Eli**$))$

Here, s is the nominal (or discourse referent) for the event **see**, which we interpret as an *observational process* in past tense. Related to the event s are two dependents: we have an Actor, a *person* with discourse referent k, being **Kathy** (the one doing the seeing); and a Patient, another *person* but with discourse referent e, being **Eli** (the one being seen).

We can flatten the representation in (1b) by rewriting it into a conjunction of elementary predications, akin to MRS terms:

(2) $@_{\{s:\mathsf{observing}\}}$(**see**)
$\&$ $@_{\{s:\mathsf{observing}\}}\langle Tense\rangle$**past**
$\&$ $@_{\{s:\mathsf{observing}\}}\langle Actor\rangle(k : person)$
$\&$ $@_{\{k:\mathsf{person}\}}$(**Kathy**)
$\&$ $@_{\{s:\mathsf{observing}\}}\langle Patient\rangle(e : person)$
$\&$ $@_{\{e:\mathsf{person}\}}$(**Eli**)

The flattened representation in (2) illustrates that we have basically three types of elementary predications: lexical predications – $@_{\{k:\mathsf{person}\}}$(**Kathy**), features – $@_{\{s:\mathsf{observing}\}}\langle Tense\rangle$**past**, and dependency relations – $@_{\{s:\mathsf{observing}\}}\langle Actor\rangle(k : person)$.

2.2 System architecture

Figure 1 describes the conceptual architecture that underlies our system. As a cognitively motivated architecture, it specifies the underlying infrastructure for the communication abilities for an intelligent embodied agent.

The distributed nature of the architecture is inspired by the general tendency to see cognition as a network of concurrent, situated processes, e.g. [Minsky, 1986; Langley and Laird, 2002]. Distributed information processing facilitates the concurrent maintenance of several models of the environment, possibly using different means for representation and

interpretation. Furthermore, we can adopt a localised approach to the processing and fusion of information stemming from different modalities (*local cross-modal fusion*), guided by passive attention mechanisms [Chum and Wolfe, 2001] and active attention mechanisms through a short-term working memory with activated concepts and their associations.

The architecture is layered in that we distinguish different levels of information processing. The levels basically correspond to the reactive (or perceptual), deliberative, and meta-level processes in the cognitive architecture presented in [Sloman, 2001]. Like cognitive robotics [Reiter, 2001] we use logic as a representational medium at the deliberative level, but with an explicit relation to lower-level perceptual processes [Shanahan, 2000; Shanahan and Witkowski, 2001].

We use context models to represent the deliberative interpretation of the situation relative to a particular modality, on the basis of which future states can be anticipated and planned. Each context model maintains a (model-specific, local) salience measure over the information in the model, to indicate what is currently activated. Examples of context models are the dialogue context model, capturing the dialogue history which serves as the background against which new dialogue moves are interpreted and planned, or the action context which keeps track of the current status of tasks and the overall action plan. Figure 1 shows these context models as *modality specific context models*. It also includes a *belief context*. The belief context model captures *global cross-modal fusion*, achieved by fusing information across the different modalities at least at the level of token identification. We establish a common ground for interpretation across modalities by relating each layer to a set of ontologies that model categories on which the events, states, and entities at that layer can be interpreted, following recent work in information fusion [Wache *et al.*, 2001] and dialogue systems [Gurevych *et al.*, 2003]. There are different levels of granularity for the common ground we may be able to establish, due to the potential for hybridity across the different local representations used in the architecture. On the low end of the scale we have type identity, to type/token identity, to the high end where we have fully shared representations. The granularity of the common ground we are able to establish determines to what extend information can be fused.

For the purposes of this paper we use the example implemented architecture, shown in Figure 2. The goal of this instance is to enable a robot to conduct a simple dialogue about a dynamic, visual scene. We have implemented the distributed infrastructure using the *Open Agent Architecture* [Cheyer and Martin, 2001]; the different boxes in Figure 2 are processes implemented as OAA agents.

On the interpretation side, we have several layers of processing for a speech dimension, and for a vision dimension. We process the acoustic signal using *Sphinx4* [Walker *et al.*, 2004] with a domain-specific, English language model, and then parse the recognized string using the *OpenCCG* parser for combinatory categorial grammar [Baldridge, 2002]. The parser yields a logical form of the meaning of the string, represented as an HLDS logical form. This logical form is interpreted further in a dialogue process, which maintains a model of the dialogue history. In parallel to the speech dimension,

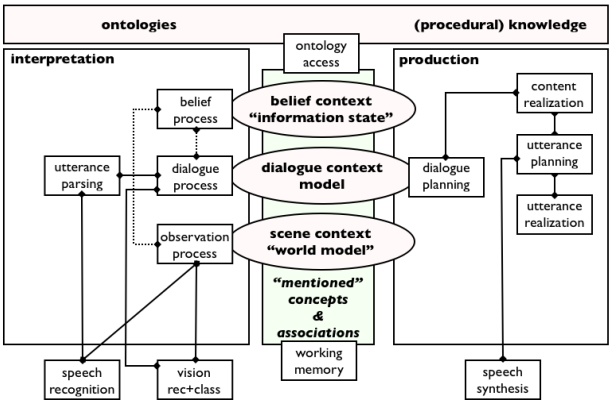

Figure 2: Example implemented architecture

we also have processes that interpret the visual scene. We use a visual recognition and classification algorithm based on *OpenCV*[1] that produces a representation of an object in terms of its type, physical properties, and position. We interpret these representations on a model of the visual scene, capturing proximal and projective spatial relations between objects [Kelleher and Kruijff, 2005b].

For production, the architecture in Figure 2 includes only spoken language as an output modality. The dialogue planner constructs an HLDS logical form that specifies the communication goal reflecting how the belief context could be updated with the information coming from the acoustic and visual dimensions. This logical form is taken by the utterance planner, which expands this logical form to a full logical form that OpenCCG can realize as a well-formed string [White, 2004]. Finally, we use FreeTTS[2] for speech synthesis.

3 Utterance planning

Following the systemic tradition, we formulate a planning grammar as a network of systems. A system represents a paradigmatic choice, i.e. a choice about an aspect of the meaning to be specified by the logical form we are planning. We specify the decision process involved in this choice as a decision tree or chooser, associated with the system. In the chooser, we can pose several inquiries about the logical form and the contextual status of discourse referents, to guide the decision process. On the basis of the choice we make, the system performs one or more operations on the logical form, to expand it; we thus reflect grammatical features directly as content in the logical form.

A system consists of an entry condition, actions associated with the different choices the associated chooser can make, and an output. Both the entry condition and the output of the system take the shape of an HLDS logical form, and an indication of the locus within that logical form. As a result, the combination of locus and output logical form of one system may be the entry condition for another system. It is in this way that we obtain a network of systems.

[1]We would like to thank Somboon Hongeng from Birmingham University for the implementation of this module.

[2]`http://freetts.sf.net`

```
<system id=''evidencing-modality'' region='''' metafunction=''ideational''>
   <chooser id=''c-evidencing-mod''/>
   <conditions>
       <condition features=''@type:process''/>
   </conditions>
   <actions>
       <action choice=''vision''>
           <assign-type type=''observing''/>
           <add-proposition propositions=''@see''/>
           <add-relation mode=''Actor'' nomvar=''sp'' type=''speaker''/>
           <identify-nomvar mode=''Patient'' nomvar=''obj''/>
           <move-locus nomvar=''obj''/>
       </action>
       :
   </actions>
</system>
```

Figure 3: Example of a system

Figure 3 provides an example specification of a system, called evidencing-modality. The point of the system is to specify the kind of mental process the current locus in the logical form should express to refer to the modality in which an entity can be grounded. The chooser associated with this system is c-evidencing-mod; cf. Figure 5 and below.

One of the possible answers of this chooser is *vision*, i.e. the visually situated context is the "strongest" modality in which we can ground the entity that is part of the communicative goal. This results in the system performing several actions on the logical form:

1. assign-type specifies the type of the locus as *observing*
2. add-proposition adds the proposition **see** to the nominal of the locus
3. add-relation adds a relation of type *Actor* between the locus and a nominal *sp* of type *speaker*
4. identify-nomvar identifies the nominal to which the *Patient* relation points, and then moves the locus to this nominal (identified by variable name *obj*).[3]

We have two more operations that a system can specify, besides the above ones. The operation add-feature adds a feature and a value to the nominal of the current locus. This operation together with the operations assign-type, add-proposition and add-relation gives us a basic inventory for extending a logical form through *substitution*:

- add-feature:
  $@_{\{n:\mathsf{nomv}\}}\phi \Longrightarrow @_{\{n:\mathsf{nomv}\}}\phi \,\&\, @_{\{n:\mathsf{nomv}\}}\langle Feat\rangle(\textbf{value})$.
- add-proposition:
  $@_{\{n:\mathsf{nomv}\}}\phi \Longrightarrow @_{\{n:\mathsf{nomv}\}}(\phi \,\&\, \textbf{prop})$
- add-relation:
  $@_{\{n:\mathsf{nomv}\}}\phi \Longrightarrow @_{\{n:\mathsf{nomv}\}}\phi \;\&\; @_{\{n:\mathsf{nomv}\}}\langle Rel\rangle n' : nomv'$.
- assign-type:
  $@_{\{n:\mathsf{nomv}\}}\phi \Longrightarrow @_{\{n:\mathsf{type}\}}\phi$

[3]We can define for a variable whether it is to have system-local scope, or global scope. This way, we can reference other parts of a logical form, outside the scope of the subtree that is currently in the locus.

Furthermore, we have an operation adjoin-lf. With this operation we can extend the current logical form by *adjoining* another logical form into it. We can explain adjunction using the illustration in Figure 4.

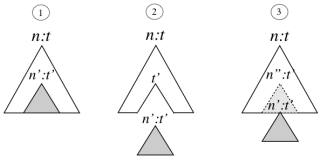

Figure 4: Adjunction

We start with a logical form which contains the greyed subtree rooted by a nominal n' of type t' (1). Next, we remove that subtree, leaving an argument position for a nominal of type t' in the logical form rooted by $n : t$, (2). We now insert a new subtree, rooted by a nominal n'' of type t', which itself also contains an argument position of type t' into which we can slot the greyed subtree of $n' : t'$. The adjunction operator, the above substitution operators, and the identify-nomvar, give us a complete inventory of operations for defining logical forms as directed graphs in HLDS.

The main way we bring in context-sensitivity is through the types of inquiries we can pose in a chooser. In the architecture, the utterance planner runs as an agent with access to the various short-term and long-term models that the robot maintains for the environment it is situated in; cf. Figure 2. Each of these models is equipped with attentional mechanisms, which model current attentional prominence (short-term working memory) or salience (longer-term memories, like models of the discursive or visual context).

Using the inquiries built into the utterance planner, we can query the architecture for the contextual status of an entity, and what the strongest evidencing modality is in which we can ground the entity.[4] Based on the results of these inquiries, we can decide how to reflect the contextual status of an entity or an event in the logical form. Contextual appropriateness can thereby take various forms, not just in terms of using information structure to reflect attentional status, but also

[4]We assume that the visual context is the strongest modality, followed by the discursive context, and then the (personal) belief context.

```
<chooser id=''c-evidencing-mod'' region='''' metafunction=''textual''>
   <dectree>
       <choicenode answer=''*TOP*''>
          <inquiry id=''fetch-evid-modality'' type=''string''
                   answerset=''@vision @dialogue @beliefs''>
              <f-mod-maxevid/>
          </inquiry>
          <choicenode answer=''vision''>
             <result val=''vision''/>
          </choicenode>
          :
        </choicenode>
    </dectree>
</chooser>
```

Figure 5: Example of a chooser

by appealing to the appropriate (and maximally informative) modal context to refer.

Another way we bring in context-sensitivity is through the inclusion of algorithms for generating referring expressions. One of the actions a system can perform is to call a dedicated GRE algorithm, to plan a contextually appropriate referring expression for an entity. Currently, the planner has access to an extension of the incremental Dale & Reiter GRE algorithm, which is able to generate a referring expression for an entity on the basis of its physical properties as well as its spatial relations to other entities in the visually situated context. The algorithm returns a full logical form for the meaning of the referring expressions. This algorithm is described in [Kelleher and Kruijff, 2005a].

Example. To illustrate the way logical forms are planned, consider the network in Figure 6. The network provides an illustration of how we could plan simple types of grounding feedback, for example to a statement about the visual scene like "The red ball is near the blue box."

Depending on whether the robot is able to verify the statement against its models of the dialogue history and the visual scene, the dialogue planner generates a simple communicative goal providing feedback to the statement. For example, if the robot is able to resolve the referents both in the dialogue and the visual contexts, then the dialogue planner will send an acknowledgment to the utterance planner: $@_{\{d:\mathsf{disc-vantagepoint}\}}\langle Acknowledgment\rangle(p1 : process)$ & $@_{\{p1:\mathsf{process}\}}\langle Patient\rangle(o1 : phys - obj)$. Here, $o1$ is an identifier for the red ball in the belief context, where we fuse the information about identifiers in the dialogue and visual context.

The utterance planner makes d the locus, and enters system (1). Here, a chooser inquires after the dialogue move to determine the polarity of the utterance. Because we need to produce an acknowledgment, the polarity is to be positive. The utterance planner now extends the logical form to express the polarity through "yes" and a positive state: We add **yes** to the nominal d, and adjoin a positive state construction between d and the process $p1$. We now get the following logical form:

(3) $@_{\{d:\mathsf{disc-vantagepoint}\}}(\textbf{yes})$
& $@_{\{d:\mathsf{disc-vantagepoint}\}}\langle Acknowl.\rangle(p1 : process)$
& $@_{\{s:\mathsf{state}\}}(\textbf{do})$
& $@_{\{s:\mathsf{state}\}}\langle Scope\rangle(p1 : process))$
& $@_{\{p1:\mathsf{process}\}}\langle Patient\rangle(o1 : phys - obj)$

Finally, we move the locus to the process $p1$, identified by the variable px in the system.

The type of $p1$, *process*, satisfies the entry condition for system (2). In this system, we inquire after the strongest evidencing modality for the entity referenced in the communicative goal. Because the strongest modality is the visual context, we turn the process into a mental process of type "observing". Although the entity can also be grounded in the dialogue and belief contexts, it would be less appropriate to say "Yes I do understand ..." or "Yes I do believe .." than it would be "Yes I do *see* ...". The remaining actions in the system add the proposition **see** to $p1$, and add an Actor relation to the speaker:

(4) $@_{\{d:\mathsf{disc-vantagepoint}\}}(\textbf{yes})$
& $@_{\{d:\mathsf{disc-vantagepoint}\}}\langle Acknowl.\rangle(p1 : process)$
& $@_{\{s:\mathsf{state}\}}(\textbf{do})$
& $@_{\{s:\mathsf{state}\}}\langle Scope\rangle(p1 : process))$
& $@_{\{p1:\mathsf{process}\}}(\textbf{see})$
& $@_{\{p1:\mathsf{process}\}}\langle Actor\rangle(sp : speaker)$
& $@_{\{p1:\mathsf{process}\}}\langle Patient\rangle(o1 : phys - obj)$

We next move the locus to Patient, i.e. the entity to be acknowledged. The type of the entity satisfies the entry condition for system (4). There we trigger the generation of a contextually appropriate referring expression, calling a GRE algorithm with the identifier $o1$.

(5) $@_{\{d:\mathsf{disc-vantagepoint}\}}(\textbf{yes})$
& $@_{\{d:\mathsf{disc-vantagepoint}\}}\langle Acknowl.\rangle(p1 : process)$
& $@_{\{s:\mathsf{state}\}}(\textbf{do})$
& $@_{\{s:\mathsf{state}\}}\langle Scope\rangle(p1 : process))$
& $@_{\{p1:\mathsf{process}\}}(\textbf{see})$
& $@_{\{p1:\mathsf{process}\}}\langle Actor\rangle(sp : speaker)$
& $@_{\{p1:\mathsf{process}\}}\langle Patient\rangle(o1 : phys - obj)$
& $@_{\{o1:\mathsf{phys-obj}\}}(\textbf{ball})$
& $@_{\{o1:\mathsf{phys-obj}\}}\langle Delimitation\rangle(\textbf{unique})$
& $@_{\{o1:\mathsf{phys-obj}\}}\langle Quantification\rangle(\textbf{specific\_singular})$

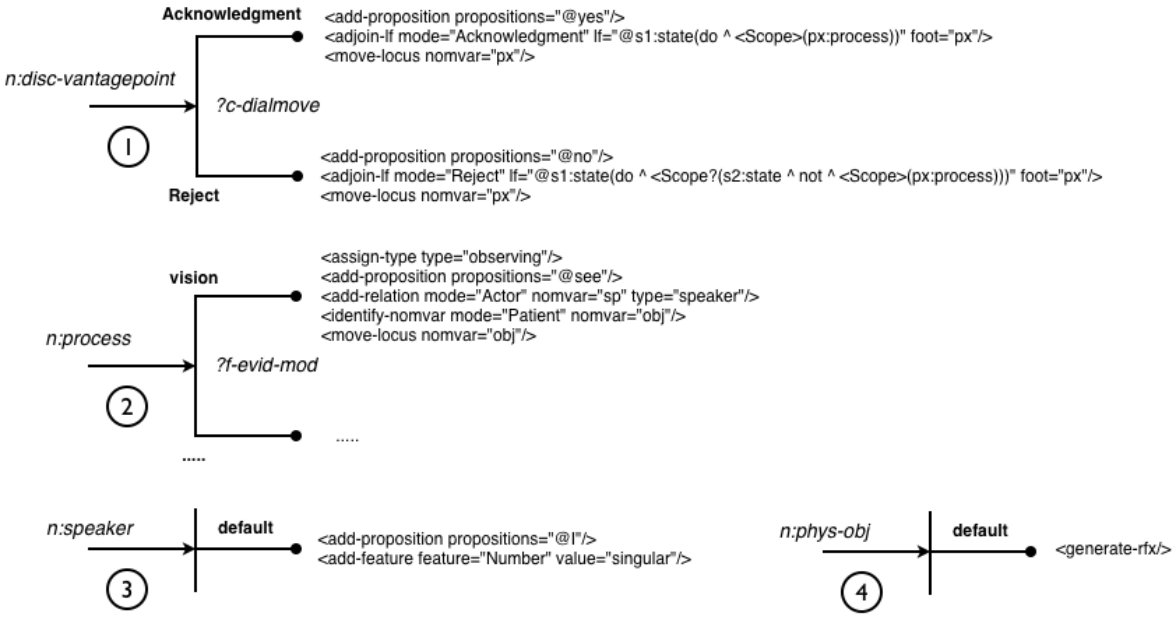

Figure 6: Simple network for planning grounding feedback

$\&\ @_{\{o1:\mathsf{phys-obj}\}}\langle Property\rangle(c1 : color)$
$\&\ @_{\{c1:\mathsf{color}\}}(\textbf{red})$

Finally, the agenda manager moves the locus automatically to the nominal sp. Applying system (3), we introduce the full specification of the speaker, to yield the logical form that is outputted by the utterance planner:

(6) $@_{\{d:\mathsf{disc-vantagepoint}\}}(\textbf{yes})$
$\&\ @_{\{d:\mathsf{disc-vantagepoint}\}}\langle Acknowl.\rangle(p1 : process)$
$\&\ @_{\{s:\mathsf{state}\}}(\textbf{do})$
$\&\ @_{\{s:\mathsf{state}\}}\langle Scope\rangle(p1 : process))$
$\&\ @_{\{p1:\mathsf{process}\}}(\textbf{see})$
$\&\ @_{\{p1:\mathsf{process}\}}\langle Actor\rangle(sp : speaker)$
$\&\ @_{\{sp:\mathsf{speaker}\}}(\textbf{I})$
$\&\ @_{\{sp:\mathsf{speaker}\}}\langle Number\rangle(\textbf{singular})$
$\&\ @_{\{p1:\mathsf{process}\}}\langle Patient\rangle(o1 : phys - obj)$
$\&\ @_{\{o1:\mathsf{phys-obj}\}}(\textbf{ball})$
$\&\ @_{\{o1:\mathsf{phys-obj}\}}\langle Delimitation\rangle(\textbf{unique})$
$\&\ @_{\{o1:\mathsf{phys-obj}\}}\langle Quantification\rangle(\textbf{specific\_singular})$
$\&\ @_{\{o1:\mathsf{phys-obj}\}}\langle Property\rangle(c1 : color)$
$\&\ @_{\{c1:\mathsf{color}\}}(\textbf{red})$

3.1 Coverage

One of the topics under investigation in the CoSy project is how a robot can *learn through language*, acquiring more knowledge of its environment through interaction with a human tutor. As such, we are currently developing grammars for the utterance planner, so as to be able to handle clarification dialogues, verbalization of what the robot does or does not know, and synchronization of different modalities (sequencing, concurrency) through which the robot can communicate. The example below illustrates such a dialogue.

(7) H: "In front of you you see a desk."
R: looks in front of it, acquires a visual recognition model of the object it has in its field of vision
R: turns to the tutor
R: nods, and says "Thank you for showing me a desk."
R: "Is a desk a kind of table?"
H: "Yes, that is correct."

4 Practical planning grammars

In this section we describe *ongoing* research on constructing utterance planning grammars for practical systems.

4.1 Derivation from a CCG grammar

In a dialogue system, there are usually various grammars: for speech recognition, parsing recognized strings, generating strings from logical forms, an utterance planning grammar; and so on. Ideally, one grammar would be enough – and should, if we want to maintain a single source of linguistic knowledge across the different levels of interpretation and production of natural language. The OpenCCG system already facilitates the use of a single grammar for both parsing and realization. Here, we discuss how we could derive the planning grammar from the signature of the CCG grammar, to ensure that we can realize what we can plan.

We start from two observations. First, we can break up the meaning of an expression, down to the level of a single word, in terms of a conjunction of elementary predications; cf. §2.1. Second, we can organize the computational CCG lexicon as a collection of (monotonic) inheritance hierarchies [Baldridge,

2002]. Taken together, this means that we can describe the hierarchies in terms of how elementary predications are added, when descending down a hierarchy. More precisely, because we can distinguish between elementary predications that add propositions, features, or relations, we can describe the hierarchies in terms of what types of structure are being added.

Hierarchies are over lexical families. Given a lexical family f_i, we can define the signature of its meaning in terms of what it (a) inherits from a super-family f_j, and (b) contributes itself: $\Sigma(f_i) = LF_{f_i} + \Sigma(f_j : f_i \sqsubseteq f_j)$. Because a contribution to logical form is essentially the specification of the type of the root nominal and a conjunction of elementary predications ep_i, LF_{f_i} is $nom : type$ plus a set of conjuncts $conj(f_i) = \{ep_1, ..., ep_n\}$. We can separate the conjuncts further in terms of relations, propositions, and features: $conj(f_i) = props(f_i)\{ep_1, .., ep_k\} \cup rels(f_i) = \{ep_l, ..., ep_m\} \cup feats(f_i) = \{ep_n, ..., ep_o\}$.

Based on the signatures, we define the construction of systems and outline their associated choosers. Staying with a functional perspective, we should not map lexical families directly onto systems. Systems define paradigmatic choices, whereas lexical families reflect an aggregation of several such choices: they reflect transitivity through their valency and the associated structure of their categories, whereas other meaningful dimensions are associated with their features.

We focus first on transitivity. This is expressed by the type of the nominal, and the elementary predications in $props$ and $rels$. Given a subtree in the inheritance hierarchy, being a family f_i and its n immediate children f_j, we can define a system σ for the transitivity region in the planning network as follows. The entry condition to the system is defined by the logical form that f_i yields, modulo its features i.e. $entry(\sigma) = \Sigma(f_i) - feats(f_i)$. The chooser needs to make $(n-1)$ decisions, to cover the different possibilities; we derive the associated actions (add-proposition, add-relation, assign-type) directly from the type for f_j, $props(f_j)$ and $rels(f_j)$. This procedure yields a shallow network, in which systems do not take into account any similarities between the children f_j of f_i. If we want this, we need to find a common structure between the children. This is again an inheritance hierarchy, over contributions to logical form in terms of $props(f_j) \cup rels(f_j)$. We then define system for each node in this hierarchy, using the above procedure.

We can thus obtain the systems for the transitivity region of the planning grammar, based on how logical forms in the lexical inheritance hierarchy are expanded by assigning more specific types, and adding propositions and relations. To organize regions around features (and their values), we suggest to let this organization follow from the organization of features and values in the type-hierarchy, which we can specify for a CCG grammar [Erkan, 2003]. Given a class of features, and a lexical inheritance hierarchy, we can define systems on the basis of how these features are set when we descend down the hierarchy. Given an inheritance hierarchy, we annotate each node for a family f_j with those features in $feats(f_j)$ that are in the class we consider. The resulting structure may be sparse, as not every family needs to add features; hence, we can flatten this structure by removing nodes that do not add any features. From this structure, we can create systems in essentially the same way as we did above. For each subtree in the structure, we create a system that has as entry condition the type for the root f_i plus its features in the current class. The chooser needs to make decisions about the specification of the features introduced or further specified by the children f_j of f_i. For both introduction and specification the associated action is add-feature; if a feature is specified, the chooser can already specify a choice point based on the inquiry after the presence and (underspecified) value of the feature.

The resulting utterance planning grammar is based on a network in which the entry conditions ensure that the way logical forms are expanded conforms to the way lexical families are formulated in the grammar. The systems are distributed across different regions, modelling different dimensions of paradigmatic choices that the systems in these regions make, whereby the organization into regions is driven by the relational structures defined through the lexical families (i.e. transitivity) and the type hierarchies over features.

Current research focuses on how we can use XSLT to transform the XSL-based specification of a CCG grammar into the different structures from which we derive the systemic network. We then use the resulting XML-based structures together with the type hierarchies for the grammar as input to the above construction procedures. The systems we can thus obtain still lack the decisions to be made in the choosers; we are developing a debugger/editor for the sentence planner to help specifying these.

4.2 Trainability

Relatively recently, the issue of trainability of utterance planners has arisen in the context of practical dialogue systems. Using training, we can automatically adapt and optimize the choices of a planner to the domain in which the planner needs to be applied. This has the potential of yielding a significantly faster planner; cf. e.g. [Stent *et al.*, 2004].

For training the planner we discuss in this paper, we are not only interested in ensuring that we obtain a logical form that is appropriate given the communicative goal to be expressed (or, in a more structured way, comparable to the rhetorical structures considered in [Stent *et al.*, 2004]); the logical form also needs to be appropriate given 'the' context.

This poses an interesting challenge, because it means that we need to train the planner on data that is rated not only for its structural appropriateness, but also for contextual appropriateness. By data we understand a domain-specific corpus of parseable expressions, annotated with context features such as the salience of an entity or event across different modalities. We are exploring how we can train the planner over the logical forms underlying the syntactic analyses of the expressions, i.e. training is not on the surface forms.

The basic idea we consider is the following. Given the logical form for an expression, written as elementary predications, we can reconstruct the path through the systemic network that gives rise to this logical form. A path consists of the systems that need to be entered, and the decisions that the associated choosers need to make. Training then comes down to learning, for each system, an n-ary classifier that

takes context features and the logical form for the current locus, to output the choice that the chooser should make.

5 Conclusions

In this paper we discussed the implementation of an utterance planner which is part of a larger communication subsystem for a conversational robot. One of the challenges for such a robot is to produce contextually appropriate utterances in a dynamic context. We presented an approach that can dynamically include information about the situated context while planning an utterance. We use systemic networks to guide the paradigmatic choices the planner needs to make, and we discussed (briefly) how these networks could be trained, and derived from the CCG grammar that specifies the linguistic knowledge of the robot. The planner is implemented in Java, and has a (`tccg`-style) debugger enabling one to trace, and interact with, the decisions the planner makes, and to realize the resulting logical forms using OpenCCG.

References

[Areces, 2000] Carlos Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. Phd thesis, University of Amsterdam, Amsterdam, the Netherlands, 2000.

[Baldridge and Kruijff, 2002] Jason Baldridge and Geert-Jan M. Kruijff. Coupling CCG and hybrid logic dependency semantics. In *Proceedings of ACL 2002*, Philadelphia, Pennsylvania, 2002.

[Baldridge, 2002] Jason Baldridge. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh, 2002.

[Bateman, 1997] John A. Bateman. Enabling technology for multilingual natural language generation: the KPML development environment. *Journal of Natural Language Engineering*, 3(1):15–55, 1997.

[Blackburn, 2000] Patrick Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Journal of the Interest Group in Pure Logic*, 8(3):339–365, 2000.

[Cassell *et al.*, 2000] Justine Cassell, Matthew Stone, and Hao Yan. Coordination and context-dependence in the generation of embodied conversation. In *Proceedings of INLG-2000*, pages pages 171–178, 2000.

[Cheyer and Martin, 2001] Adam Cheyer and David Martin. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1):143–148, March 2001.

[Chum and Wolfe, 2001] M. Chum and J. Wolfe. Visual attention. In E. Bruce Goldstein, editor, *Blackwell Handbook of Perception*, Handbooks of Experimental Psychology, chapter 9, pages 272–310. Blackwell, 2001.

[Copestake *et al.*, 1997] Ann Copestake, Dan Flickinger, and Ivan A. Sag. Minimal recursion semantics. an introduction. Unpublished Manuscript. CSLI/Stanford University, 1997.

[Erkan, 2003] Güneş Erkan. A type system for CCG. Master's thesis, Middle East Technical University, Ankara, Turkey, 2003.

[Foster and White, 2004] Mary Ellen Foster and Michael White. Techniques for text planning with XSLT. In *Proceeedings of NLPXML-2004*, Barcelona, Spain, 2004.

[Gurevych *et al.*, 2003] Iryna Gurevych, Robert Porzel, Elena Slinko, Norbert Pfleger, Jan Alexandersson, and Stefan Merten. Less is more: Using a single knowledge representation in dialogue systems. In *Proceedings of the HLT-NAACL WS on Text Meaning*, Edmonton, Canada, 2003.

[Hobbs, 1985] Jerry R. Hobbs. Ontological promiscuity. In *Proceedings of ACL 1985*, 1985.

[Kelleher and Kruijff, 2005a] John D. Kelleher and Geert-Jan M. Kruijff. A context-dependent algorithm for generating locative expressions in physically situated environments. In *Proceedings of ENLG-05*, Aberdeen, Scotland, 2005.

[Kelleher and Kruijff, 2005b] John D. Kelleher and Geert-Jan M. Kruijff. A context-dependent model of proximity in physically situated environments. In *Proceedings of the ACL-SIGSEM workshop The Linguistic Dimension of Prepositions*, Colchester, England, 2005.

[Kruijff, 2001] Geert-Jan M. Kruijff. *A Categorial-Modal Logical Architecture of Informativity: Dependency Grammar Logic & Information Structure*. PhD thesis, Charles University, Prague, Czech Republic, 2001.

[Langley and Laird, 2002] Pat Langley and John E. Laird. Cognitive architectures: Research issues and challenges. Technical report, Institute for the Study of Learning and Expertise, Palo Alto, CA, 2002.

[Mathiessen, 1983] Christian M.I.M. Mathiessen. Systemic grammar in computation: the Nigel case. In *Proceedings of EACL 1983*, 1983.

[Minsky, 1986] Marvin L. Minsky. *The Society of Mind*. Simon and Schuster, New York, NY, 1986.

[Reiter, 2001] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Cambridge MA, 2001.

[Shanahan and Witkowski, 2001] Murray P. Shanahan and Michael Witkowski. High-level robot control through logic. In *Intelligent Agents VII*, pages 104–121. Springer-Verlag, Berlin, Germany, 2001.

[Shanahan, 2000] Murray P. Shanahan. Reinventing Shakey. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 233–253. Kluwer Academic Publishers, Dordrecht, the Netherlands, 2000.

[Sloman, 2001] Aaron Sloman. Beyond shallow models of emotion. *Cognitive Processing*, 2(1):177–198, 2001.

[Stent *et al.*, 2004] Amanda Stent, Rashmi Prasad, and Marilyn Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of ACL 2004*, Barcelona, Spain, 2004.

[Stone and Doran, 1997] Matthew Stone and Christine Doran. Sentence planning as description using tree-adjoining grammar. In *Proceedings of ACL 1997*, pages 198–205, 1997.

[Wache *et al.*, 2001] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of IJCAI 2001 Workshop "Ontologies and Information Sharing"*, Seattle WA, 2001.

[Walker *et al.*, 2004] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. Technical report, SUN Microsystems Inc., 2004. Technical Report TR2004-0811.

[White and Baldridge, 2003] Michael White and Jason Baldridge. Adapting chart realization to CCG. In *Proceedings of ENLG-03*, Budapest, Hungary, 2003.

[White, 2004] Michael White. Efficient realizations of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 2004.