NECA

NET ENVIRONMENT FOR EMBODIED EMOTIONAL CONVERSATIONAL AGENTS

D3a: Specification of Scene Descriptions for the NECA Domains

Paul Piwek, ITRI

Version: 2.0

Date: April 11, 2002

Project ref. No.	*IST-2000-28580*
Project title	**NECA: A Net Environment for Embodied Emotional Conversational Agents**
Deliverable status	**Internal**
Contractual date of delivery	*March 31, 2002*
Actual date of delivery	*April 11, 2002*
Deliverable number	*D3a*
Deliverable title	Specification of Scene Descriptions for the NECA Domains
Type	Report
Status & version	Final
Number of pages	31
WP contributing to the deliverable	WP 3
Task responsible	UoB
Author(s)	Paul Piwek
EC Project Officer	Patrice Husson
Keywords	Network Representations, Scene Descriptions, Semantics, Pragmatics, Dialogue Acts, Description Logic
Abstract (for dissemination)	Specification of preliminary formalisation of domain knowledge for the Socialite and the eShowRoom scenarios.

ITRI – University of Brighton

Watts Building

Moulsecoomb, Brighton BN2 4GJ

UK

Email: Paul.Piwek@itri.bton.ac.uk

Contents

Executive Summary

The objective of the NECA project is to develop a new, more sophisticated generation of conversational characters: on-line beings that are able to speak and act like humans. The project focuses on communication between animated characters that exhibit credible personality traits and affective behaviour. Two concrete application scenarios will be developed, each of which enhances products that are already on the market. The first scenario, 'eShowRoom' concerns sales dialogues between a seller and one or more buyers. The second scenario 'Socialite' uses the same technology for a different purpose. Users can create characters that they send into a virtual world where they will meet and interact with other agents.

The NECA system generates the interaction between two or more characters in a number of steps. Following terminology that is common from the theatre, we call such an interaction a scene. At first, a characterization of the scene is generated which describes the content, type and emotion of the actions that the agents engage in, a so-called Scene Description. This Scene Description is then translated by a Multi-Modal Natural Language Generation module, a Speech Generator and an Animation Generator into a script (for spoken words, gestures, facial expressions and other actions) that can be performed by a group of animated characters. This document provides a specification of the Scene Descriptions.

Purpose

In the NECA System, Scene descriptions are the messages that are passed on from the Scene Generator to the Multi-Modal Natural Language Generator. The aim of this document is to provide a first specification of the content of the scene descriptions for the NECA domains (i.e., eShowRoom and Socialite). The problem is addressed in two steps. First, we propose a number of requirements that Scene Descriptions should satisfy. These are derived from the nature of the application domains, the requirements of the modules that manipulate the Scene Descriptions and the needs of the system developers who have to build and maintain the algorithms and resources for processing the Scene Descriptions. The presentation of the requirements is followed by a specification of a representation format that meets these requirements. A simple network representation is used for this purpose. The application of this representation format to the eShowRoom domain is described in some detail.

Introduction

The NECA system generates scripts that are enacted by two or more animated characters. The creation of these scripts proceeds in a number of stages. Here we concentrate on the initial stages. On the basis of a database containing information about the subject matter of the interaction, a role assignment to the characters and a specification of their personality, a so-called scene generator produces an abstract description of the interaction, henceforth the Scene Description (SD). This SD is handed over in its entirety to the Multimodal Natural Language Generation module (M-NLG). At the point where it is handed over, it will be in the *form* of an XML document (see NECA D5a). The M-NLG turns the SD into a set of instructions (in XML format) for the Speech and Animation Generators. Our concern is with the content of the messages passed from the SG to the M-NLG. It is assumed that the SG produces a complete SD and then hands it over. This approach avoids interleaved processing by the SG and the M-NLG.

We use the term scene here in the following sense:

> "**I** Theatr. **1** A subdivision of (an act of) a play, in which the time is continuous and the setting fixed, marked in classic drama by the entrance or departure of one or more actors and in non-classic drama often by a change of setting; the action and dialogue comprised in any one of these subdivisions." (source: Electronic New Shorter Oxford English Dictionary, 1996)

A scene description consists of a formal representation of a scene. It lays down (1) which participants are involved in the scene and their properties (e.g., personality), (2) what information constitutes the common ground of these characters at the beginning of the scene, (3) the acts which they are involved in (communicative and non-communicative ones) and (4) the temporal ordering of these acts. The exact description of communicative acts, henceforth dialogue acts, is undertaken below. For the moment, it is important to note that they are described primarily in terms of the *type* of act and the *informational content* of the act (along the lines of Searle's (1969) distinction between the force and the propositional content of a

speech act).[1] The Scene Description does not contain the *form* of the acts. It is up to the M-NLG and subsequent modules to compute the linguistic and extra-linguistic form (e.g., gestures and facial expression). In addition to *type* and *informational content* of the acts, the M-NLG is also supplied (amongst other things) with the *emotion* with which the act in question is carried out.

In the remainder of this document we proceed as follows. We start by presenting a list of requirements which scene descriptions should satisfy. We then provide an informal description of simple network representations which meet the aforementioned requirements. The informal description is followed by an explicit formal specification. We conclude with a worked out example of a scene description and a discussion of further (open) issues.

Requirements

The requirements for the descriptions that we propose here are derived from three sources:

1. **The application domains**:

    a. *Ability to represent combinations of different types of information*. In our application domains the scene descriptions contain representations of dialogues between two or more characters. The representation of these dialogues involves various types of information, e.g., semantic, pragmatic and affective information;

    b. *Expressive power required by the domain*. Our representations will include representations of the semantic/informational content of dialogue acts. Current semantic representations are specified using context-free grammars (which coincide with the expressive power of BNF notation). Examples of semantic representation languages are Predicate Logic (see, e.g., Partee et al., 1990) and Discourse Representation Theory (Kamp & Reyle, 1993). In order for our representation to include expressions of such semantic representation languages it should be able to incorporate representations that are generated by means of context-free grammars.

2. **The task of the modules which manipulate the descriptions**;

    a. *Ease of manipulation*. Algorithms for changing the representations should be easy to write. Ideally, incremental construction of the representations should be supported.

[1] Note that we use the term *informational* content instead of Searle's *propostitional* content. The reason for this is that the content of some speech acts is not a propositional content. In particular, the content of WH-questions (such as, 'What is the price of this car?') is not a proposition. Rather, it is thought to be a propositional function, that is a function from, in this case, prices to propositions. The idea is due to Cohen (1929) (*cf.* Bäuerle & Zimmermann 1991). The propositional function associated with the WH-question 'What is the price of this car?' takes as its value a price (the answer) , say £5000,- and returns the proposition *the price of this car is £5000,-*. Such an answer is a *true* answer if the resulting proposition is true.

b. *Ease of search*. The representations should allow for simple algorithms to search for items in the representations.

3. **The developers of the system who need to implement and maintain algorithms and resources for manipulating the descriptions**.

   a. *Predictability of representations*. Once one bit of the representation has been understood, it should be easy to predict how other parts of the representation will look like.

   b. *Locality of the specifications*. Changes to one part of the specification of a representation should not require changes to other parts of the specification, or at least as few as possible. In other words, the specification should be as modular as possible. (E.g., the introduction of new attribute of an object of type *A* should not affect attributes of other objects that take objects of type *A* as their value).

   c. *Conciseness of specifications and representations*. This applies to both the actual representations and their specification. The specification of the descriptions should be as short as possible. The representations themselves should be as short as possible as well.

   d. *Intelligibility of representations*. Wherever possible informative names and abbreviations should be used.

Informal Specification and Example

What are networks good for?

In order to satisfy as many of the requirements given above as possible, we have chosen to use *network representations* for Scene Descriptions. Networks have the following features.

1. Networks are:

   a. *Uniform*. They allow for the uniform representation of different types of information

   b. *Expressive*. They can be used to encode structures generated by context-free grammars.

2. Networks are *flat* representations which allow for:

   a. *Incremental* extension,

   b. Easy *inspection*.

3. The *uniformity* of networks supports their predictability. They can be specified in a concise manner using the *description logical notions of a T-Box and an A-Box* (see the pages 18 to 24).

What is a network?

Roughly speaking a network is a collection of labelled nodes that are connected by labelled arrows (directed arcs). Nodes stand for simple or complex objects. We discern three types of nodes:

1. Nodes that stand for arbitrary objects. Such a node conveys the existence of an object, but does not refer to a specific object. We call a node of this sort a **variable**.

2. Nodes that stand for specific named objects. Nodes of this sort are called **constants**.

3. **Complex Nodes**. An example of a complex node is, for instance, a node that represents a *set* or *list* of nodes of category 1. or 2.[2]

Nodes are labelled. The label of a node conveys the *type* of the object(s) that the node represents. Finally, nodes can be connected to each other by means of labelled arrows. If one node, say v_1, is connected to a second node, say v_2, by means of an arrow labelled $Attr_1$, we interpret this as meaning that the *value* of the *attribute* $Attr_1$ of node v_1 is equal to v_2. See Figure 1.

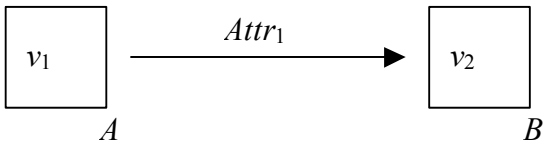

Figure 1

In Figure 1 nodes are represented by means of boxes. In this particular figure, there are two nodes, v_1 and v_2. The type of v_1 is A . The type of v_2 is B. The attribute $Attr_1$ of v_1 has the value v_2. We use the letters v_1, v_2,… for variable nodes.

A Network Representation of an eShowRoom Scene

Let us now work through a scene description from top to bottom (our example is taken from the eShowRoom car sales domain). At the root of the network (Figure 2) there is a node v_1 representing the scene itself. The scene has three attributes whose values are: 1. the set of persons who are participants in the scene, 2. a Discourse Representation Structure (DRS) which represents the common ground amongst the interlocutors at the start of the

[2] Strictly speaking we do not need complex nodes. It is possible to encode sets, lists and other types of complex nodes in terms of simple nodes. However, such representations can become very unwieldy. They would violate the requirement that our representations should be as concise as possible.

conversation, and 3. a history which consists of a set of temporal relation statements and a set of acts.

The network which is depicted in Figure 2 is only part of the entire Scene Description: the attributes of the lower three nodes are not included. We proceed with our overview of the Scene Description by looking at those nodes and their attributes.

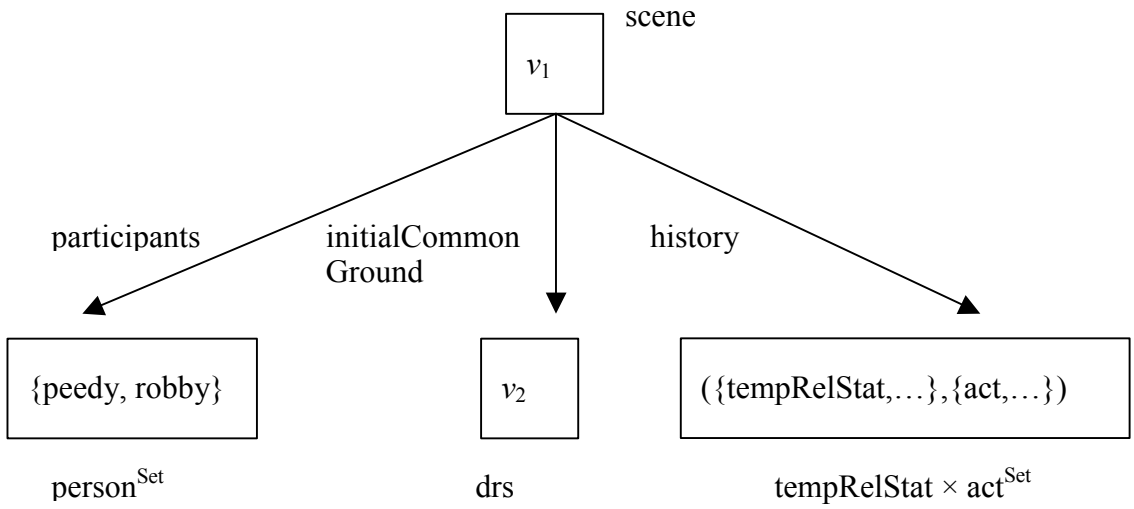

Figure 2

In this particular scene, the value of the participants attribute is a set containing two characters (persons): peedy and robby. Peedy and robby are both objects in their own right. Both objects have the attributes role and personality. In the eShowRoom scenario there are two possible roles: buyer and seller. The personality is analysed in terms of attributes such as: extraversion, agreeableness and neuroticism. Note that the objects peedy and robby are constants. They refer to specific individuals. The attributes of peedy can be represended as follows in the Scene Description:

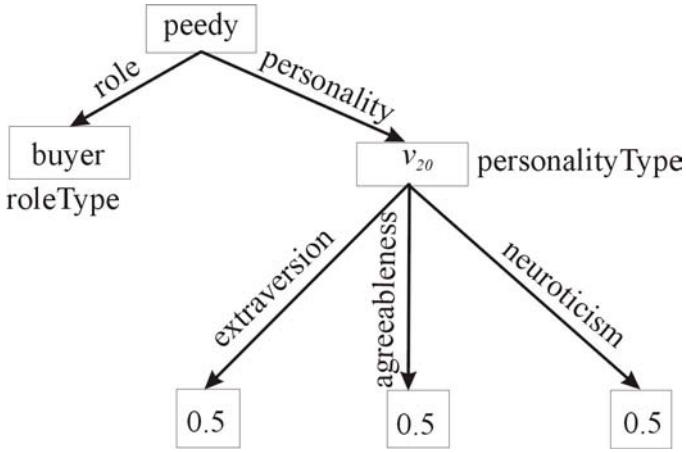

Figure 3

The initialCommonGround of the participants contains the information which is common amongst the participants at the beginning of the scene. In the course of the scene, (dialogue) acts by the participants will lead to updates of the common ground. These updated versions of the common ground are, however, not represented in the Scene Descriptions. Rather, the M-NLG keeps track of the common ground (i.e., it updates it) when it generates the form of the acts which are part of the scene. For this purpose, it has a module internal store. For a discussion of further issues relating to the representation of the common ground see the section "Representing the Common Ground" on page 25.

The next question we want to address is that of how to formally represent the content of the common ground. We propose to use a well-tested theory for the representation of discourse information: Discourse Representation Theory (DRT; Kamp & Reyle, 1993). The basic structures in DRT are Discourse Representation Structures (DRSs). A DRS consists of two parts:

1. A set of discourse referents.
2. A set of conditions.

DRT was originally devised to explain how the reader of a text builds up an internal representation of the semantic content/meaning of that text. In NECA we borrow some of the notions of DRT and use them to model the inverse of the aforementioned process: the generation of text (that is text marked up with further information on prosody, gestures, etc.) from representations of semantic content. For expository reasons, let us, however, first look at the original use of DRT.

According to DRT, the interpreter of a text incrementally builds up a representation of the content of a text in the form of a DRS. Roughly speaking, different expressions contribute different information to the DRS. For instance, consider the following little text:

Example 1 A car has 80 hp. It/The car costs 20.000 Euro.

An indefinite noun phrase, such as "a car" is assumed to introduce a *new* discourse referent, say *x*. It also causes a condition to be introduced. The condition in question is *car(x)*. The verb phrase "has 80 hp" introduces only a further condition: *has(x,80hp)*. The result of processing the first sentence of our example discourse is therefore:

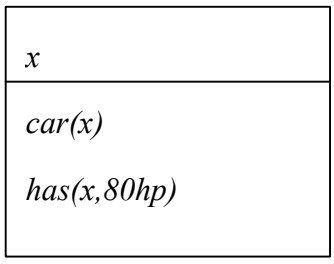

Figure 4

Here the upper part of the box is reserved for discourse referents and the lower part contains conditions. Usually conditions introduce properties of discourse referents. In the second sentence of Example 1 we encounter a definite noun phrase: "It" (a pronoun) or "The car" (a

definite description). Such a definite noun phrase often contributes no new information at all: it is used primarily to identify a discourse referent which was introduced earlier on, in this case the referent *x*. The verb phrase introduces a further condition on the referent: *cost(x,20.000Euro)*.

Note that this example is intended only for illustrative purposes. It is not meant to commit us to any specific representation of predicates and their arguments in DRT. In particular, a condition such as *cost(x, 20.000Euro)* could also have been represented by means of the following condition: *attribute(x,price,20.000)*. In our formal specification, we have adopted conditions of this form, i.e., *attribute(object,attribute_name,attribute_value*). After processing the entire text we therefore end up with the following Discourse Representation Structure:

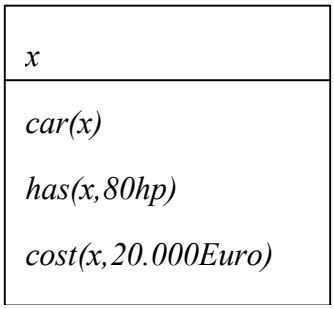

Figure 5

The encoding of such DRSs in a network representation is rather straightforward. We simply reify the notions of a DRS, referent and condition. So we now have objects of the type drs, condition and referent. Objects of the type drs have an attribute refCond whose value is pair consisting of a set of referents and a set of conditions. Each condition is again an object in its own right. We discern various subtypes of the type condition: unaryCondition, binaryCondition, ternaryCondition, etc. That is, we have conditions with one, two, three, … arguments. A unaryCondition has two attributes: one for its predicate called pred and one for its argument called arg. The value of pred is a unary predicate and the value of arg is a term (both referents and drtConstants, such as '80hp', are terms). Similarly, a binaryCondition has three attributes called pred, argOne and argTwo. Pred has a binary condition as its value and argOne and argTwo have terms as their values. Generally, n-ary conditions are treated along these lines.

The DRS in Figure 4 can now be represented by means of the network in Figure 6.

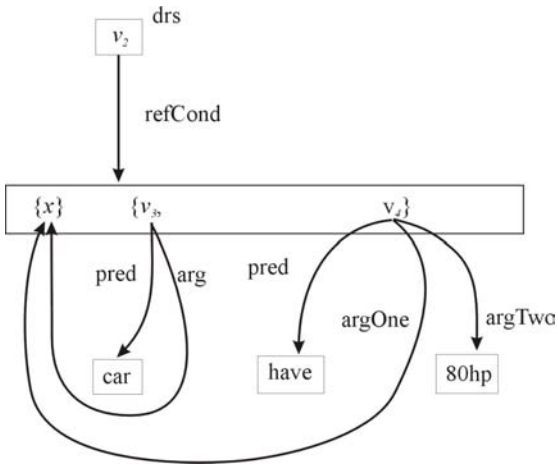

Figure 6

Let us now proceed to consider the fourth and last attribute of a scene: the history. The history consists of a set of acts and a set of statements about the temporal relations between the aforementioned acts. We have chosen *not* to represent the history as an ordered list of acts (where the ordering conveys the temporal order of the acts). We propose to represent the set of acts and their temporal order separately. Such a representation has some advantages over a simple list representation. It allows us to represent underspecified temporal orderings of the acts. What is more, it allows for incremental extensions of underspecified orderings. We need underspecification because the scene generator might not always be able to determine the order of two acts. For instance, consider the following two utterances: "The car has 80 hp" and "The car is a Mercedes". The Scene generator might not have the information to decide in which order these should be presented. In the current approach it could leave this decision to the Multimodal Natural Language Generator, i.e., it could pass a partially ordered history to the M-NLG.

So let us have a look at a possible value of the history attribute:

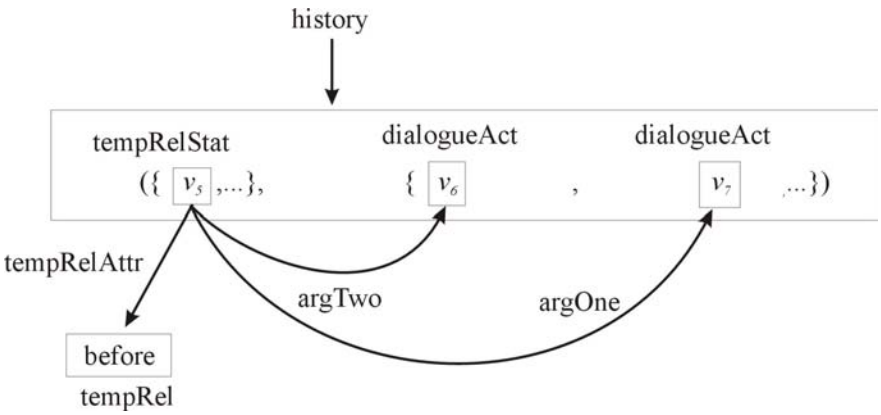

Figure 7

Here we have a set of dialogue acts which contains amongst others the dialogue acts v_6 and v_7. We also have a set of temporal relation statements. One of them is v_5. It says that v_7 precedes v_6 in the temporal ordering. To complete our top-down descent through a scene, let us look at a dialogue act (Figure 8). It is important to note that the term dialogue act here is used to cover any communicative act in dialogue. Such an act need not be expressed by linguistic means. For instance, a 'negative response' dialogue act can be realized by saying 'no' but also, for instance, by simply by shaking the head (choice of a particular gesture is of course subject to variation across different cultures).

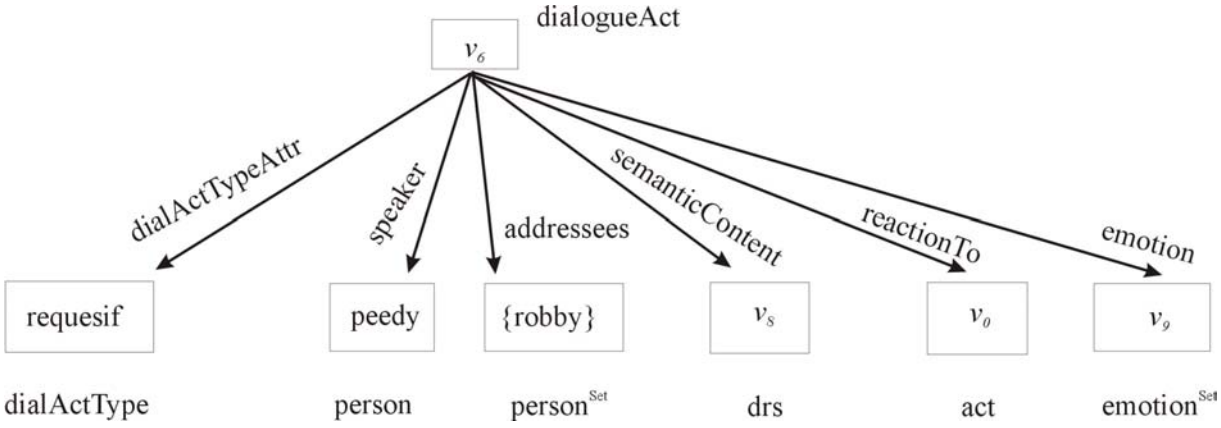

Figure 8

Emotions and DRSs have themselves again attributes. For instance, an emotion which is associated with a dialogue act can have attributes for its emotion type, intensity and the object at which the emotion is directed. We have already seen an example of a DRS. Here, let us consider a DRS which represents the semantic content of the following utterance "Does it have leather seats?/Does the car have leather seats?". For that purpose, the object v_8 would need to look as follows:

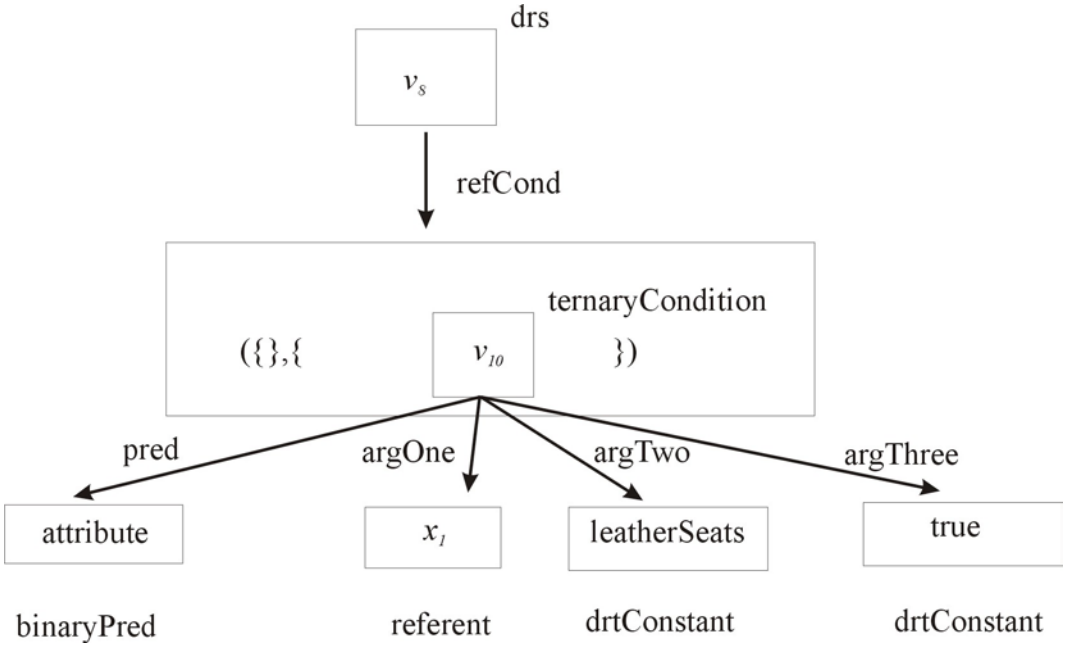

Figure 9

Encoding Semantic Content

We have chosen to encode the semantic content of dialogue acts using a network. In this network we emulate Discourse Representation Structures. Of course we could also have chosen for a more hybrid approach, where representations of the network are combined with genuine Discourse Representation Structures. This would have allowed us to write down the semantic content of dialogue acts in a more concise way. There are, however, at least two good reasons for emulating Discourse Representation Structures in the Network:

1. *Uniformity of representation*. Our approach allows us to use one single formalism to represent the entire scene. This is not only to be preferred for esthetical reasons, but also from a practical point view. For instance, it allows us to devise a uniform procedure from transferring our scene descriptions into XML format. What is more, it allows us to point from one part of the scene description into a different part of the scene description in a straightforward manner (if we had different representational languages, we would need special terms to link expressions from different languages with each other).

2. *Availability of handles*. The representation of Discourse Representation Structures (DRSs) in the network has more structure than simple DRSs. We now have objects representing DRSs and conditions. This means that it becomes easier to refer to the parts of a DRS. The M-NLG might use this additional expressive power to internally represent that a particular syntactic unit is expressing a specific part of the semantic content.

Having said this, the next few examples of semantic content will be represented in the standard DRT notation. The representations are, however, to be understood as mere abbreviations for the sake of improving the readability.

The representation of (affective) attitudes

Consider "This beautiful car has 80hp". For the moment we assume that there are at least two possible semantic representations from which this sentence could have been generated. Firstly, the predication that the car is beautiful could be part of the semantic content (we assume that the referent for the car itself, i.e., x_1, has been introduced in the initial common ground):

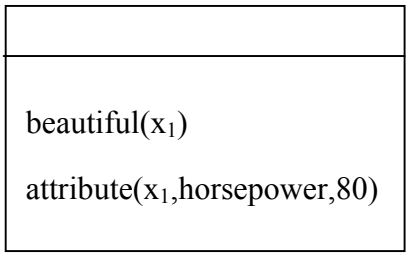

Figure 10

Note that this semantic content could also have been realized as "This car is beautiful. It has 80hp". The alternative representation is the following one (again we have not represented the DRS in full detail):

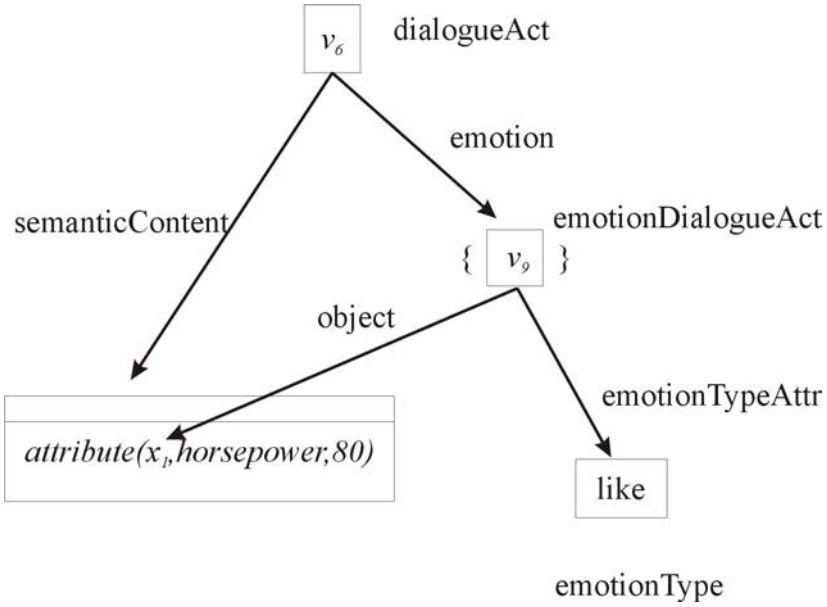

Figure 11

WH-Questions

An example of a WH-question is: “What is the price of this car?” We assume that the semantic content is represented by means of DRS. A meta-condition (in italics) is used to indicate which variable in the representation is free. The idea is that a WH-question correspond with an open proposition, i.e., a propositional function (see footnote 1 on page 6).

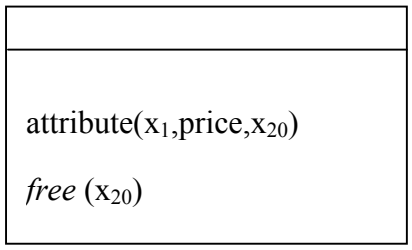

Figure 12

The type of the dialogue act in question would be a ‘requestValue’.

Open-ended Questions

Some questions allow for a wide range of possible answers. Roughly speaking, these questions introduce a discourse topic and solicit information about the topic from the addressee. We will assume that the dialogueActType in question is ‘requestInfo’. The content is represented by means of a meta-condition. Consider, for example , the semantic content of the question “What about this car/Tell me more about this car”.

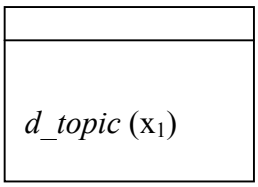

Figure 13

We assume that x_1 has been introduced as a car in the initial common ground.

Bridging Anaphora

Compare (1) “What about this car?” with (2) “What about its interior?” The car mentioned in sentence (1) was already part of the common ground of the interlocutors before the sentence was uttered. This makes it possible to refer to the car. Sentence (2) exhibit a related but more complicated phenomenon. In (2), reference is made to the interior of the car. In other words, the interior of the car is presupposed, it is assumed to be already part of the the common ground. The interior might be present only in an indirect manner: a car is part of the common ground and it is also common ground that cars normally have an interior; therefore, the interior of the car is implicit in the common ground. It is, however, not evident that we want

to invoke such a solution which involved populating the common ground with default rules (e.g., cars normally have an interior, etc.).

Here let us consider a practical solution for this problem. We represent the (presupposed) information that the car has an interior as a meta-condition in the semantic content of the dialogue act. The meta-conditions for representing such information will be called 'bridges' (between the common ground and the content of the utterance). For an example see Figure 14.

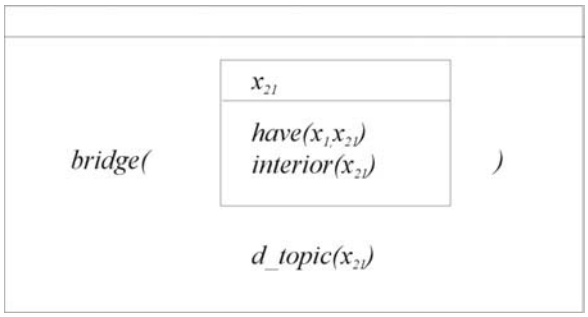

Figure 14

Formal Specification

In the preceding section we have given an example of a concrete network representation of a Scene Description. In this section we provide the precise specification of such representations. We follow the common practice in Description Logics where a distinction is made between a so-called T-Box and an A-Box (the Terminological and the Assertion box, respectively). Individual networks are represented in the A-Box. The T-Box provides domain-dependent constraints on the form of such networks. Our domain is that of Scene Descriptions. Our T-Box therefore defines the notion of a Scene Description. As we shall see, within the T-Box we can also distinguish between further subdomains (such as eShowRoom and Socialite).

Before we specify our T-Box for Scene Descriptions, let us first specify in more general terms what we consider to be a T-Box.

T-Box: Generic Specification

A T-Box consists of the following elements:

- A set of variables V;
- A set of constants C;
- A set of types T;
- A set of complex types CT which is defined by the following BNF rule: $CT ::= T \mid CT^{Set} \mid CT^{List} \mid CT \times CT$. This definition allows us make, for instance, from the simple type A a type A^{Set}. The interpretation is that the instances of A^{Set} are sets of instances of A. Similary, we can, for instance, create the type $A \times B$ whose instances are pairs of instances (a,b) such that the type of a is A and the type of b is B;
- The function $subtype$ from types to the set of their subtypes (the range of this function is T);
- The function $attributes$ from the types in T to sets of pairs consisting of an attribute name and the type (from CT) of the values which are allowed for that attribute. We write '(attribute_name, type_of_value)'. Thus for a given type A we might have $attributes(A) = \{(attr\_name1, type\_name1), (attr\_name2, type\_name2^{Set}), \ldots\}$.
- The functions $constants_D$ from types in T to the constants which can be used to refer to inhabitants of T given the (sub)domain D. For the moment we consider the domains eShowRoom (eSR) and Socialite (SCL). We stipulate that only for types which have no subtypes the function $constants_D$ can return a non-empty set. Furthermore, the sets of constants of different types are disjoint.

There are further formal constraints which a T-Box should satisfy. In particular, the type hierarchy which is induced by the subtype function should be non-circular.

T-Box for Scene Descriptions

To obtain a specific T-Box we need to fix the sets of variables V, constants C, types T and the functions subtype, attributes and constants_D. We assume that V is a denumerable set $\{v_1, v_2, \ldots\}$. Formally, C and T need to be introduced separately. They can, however, be inferred from our definition of subtype, attributes and constants (i.e., assume that there are no types and constants other than the ones mentioned in the specification below).

attributes(scene) = { (participants, $\text{person}^{\text{Set}}$),
(initialCommonGround, drs),
(history, $\text{tempRelStat}^{\text{Set}} \times \text{act}^{\text{Set}}$) }

A scene is defined as having three attributes: participants , initialCommonGround and history. The type of the values of these attributes is also specified.

attributes (person) = { (personality, personalityType),
(role, roleType) }

attributes (personalityType) = { (extraversion,zeroToOneScale)
(agreeableness,zeroToOneScale),
(neuroticism, zeroToOneScale)}

$\text{constants}_{\text{eSR/SCL}}$(zeroToOneScale) = *Constants for decimal numbers between 0.0 and 1.0*

$\text{constants}_{\text{eSR}}$(roleType) = { buyer, seller }

$\text{constants}_{\text{SCL}}$(roleType) = { userAvatar, systemAvatar}

We stipulate the persons have a personality and a role attribute. The personality consists of three attributes: extraversion, agreeableness and neurotism. The value of the role attribute is domain dependent. For the eShowRoom domain it can be buyer or seller. For the Socialite domain we distinguish between user-defined avatars (userAvatar) and avatars which are fully specified by the application designer (systemAvatar).

attributes(tempRelStat) = { (tempRelAttr, tempRel),
(argOne, act),
(argTwo, act)}

constants(tempRel) = { before, simultaneous }

Temporal relations between acts are expressed by simple expressions stating whether one act is simultaneous with another one, or whether one act precedes another one. Note that here we are describing the temporal relations which are dictated by the Scene Generator. More fine-grained relations concerning, for instance, the timing relations between words and gestures are not yet determined. These are the responsibility of the Multimodal Natural Language Generator. It determines on the wording and gestures with which dialogue acts are realized and the temporal relations between them.

subtypes(act) = { dialogueAct, nonCommAct }

attributes (dialogueAct) = { (dialActTypeAttr, dialActType),
(speaker, person),
(addressees, person^{Set}),
(semanticContent, drs),
(reactionTo, act),
(emotion, emotion^{Set}) }

$constants_{eSR}$(person) = { peedy, robby, … }

$constants_{SCL}$(person) = { maria, robert, … }

$constants_{eSR/SCL}$(dialActType) = { greeting,
openingQuestion,
openingResponse,
expressDoubt,
requestInfo,
positiveEvaluation,
negativeEvaluation,
informNeg,
informPos,
informValue,
negativeResponse,
responseNegativeResponse,
requestIf,

feedback,

initiateClosing,

completeClosing}

The following are examples of dialogue acts in the eShowRoom domain. They have been analysed in terms of the aforementioned dialogue acts:

- *greeting* *Welcome! My name is Peedy.*
- *openingQuestion* *How can I help you?*
- *openingResponse* *Could you tell us something about this car?*[3]
- *ExpressDoubt* *Really? I can't believe it! | You're kidding!*
- *requestInfo* *Tell me more about its interior!*
- *positiveEvaluation* *This is a very sporty car.*
- *NegativeEvaluation* *This is a rather expensive car.*
- *informPos* *It consumes 8 litres per 100 km*
- *negativeResponse* *I'm worrying about the running costs.*
- *responseNegativeResponse* *Forget the running costs. Think of the prestige.*
- *feedback* *LookConfused*
- *requestIf* *Does it have leather seats?*
- *initiateClosing* *This is not what we are looking for.*
- *completeClosing* *Well, nothing to be done*

subtypes(emotion) = { emotionDialogueAct,

emotionPerson }

We discern two types of emotions. The emotion with which a dialogue act is to be uttered (emotionDialogueAct) and the emotion of a particular person/character (emotionPerson). Dialogue acts have an attribute "emotion". The value of this attribute is a set of emotions. For instance, a specific dialogue act might be associated along these lines with three emotions: (1) an emotion with which the dialogue act is to be executed (the emotionDialogueAct)(2) the emotion of the speaker during the

[3] Note that this one could alternatively have been analysed as a requestInfo.

executing of the Dialogue Act (emotionPerson) and (3) the emotion of the addressee of the Dialogue Act during the performance of the dialogue act (emotionPerson).

attributes(emotionDialogueAct)	=	{ (emotionTypeAttr, emotionType), (intensity, intensityValue), (object, referent) }
attributes(emotionPerson)	=	{ (emotionTypeAttr, emotionType), (intensity, intensityValue), (object, referent), (emotionPerson, person) }
$\text{constants}_{\text{eSR/SCL}}$(emotionType)	=	{ like, dislike, … } [4]
$\text{constants}_{\text{eSR/SCL}}$ (intensityValue)	=	*constants for numeric values (from 0.0 to 1.0 with 0.5 as the default)*
attributes (drs)	=	{ (refCond, $\text{referent}^{\text{Set}} \times \text{condition}^{\text{Set}}$) }
constants(referent)	=	{ $x_1, x_2, \ldots$ }
subtypes(condition)	=	{ unaryCond, binaryCond, …, negation, conditional, disjunction, …, metaCond }
subtypes(term)	=	{ drtConstant, referent }
attributes (unaryCond)	=	{ (pred,unaryPred), (argOne, term) }
attributes (binaryCond)	=	{ (pred, binaryPred), (argOne, term), (argTwo, term) }
…	…	…

[4] For further emotion types see, for instance, Ortony, Clore and Collins (1988).

attributes (negation) = { (argOne, drs) }

attributes (conditional) = { (argOne, drs), (argTwo, drs) }

… … …

Here we have defined Discourse Representation Structures. Note that drss and conditions are treated as objects with attributes.

subtypes(metaCond) = { dTopic, bridge, free}

attributes (dTopic) = { (argOne, term) }

attributes (free) = { (argOne, referent) }

attributes (bridge) = { (argOne, drs) }

dTopic, bridge and free are defined as metaconditions. Note that metacondition is a subtype of condition. Therefore metaconditions can appear inside drss.

$\text{constants}_{\text{eSR}}$(unaryPred) = { car, … }

$\text{constants}_{\text{SCL}}$(unaryPred) = { maria, … }

… … …

$\text{constants}_{\text{eSR/SCL}}$(ternaryPred) = { attribute, …}

… … …

$\text{constants}_{\text{eSR}}$(drtConstant) = { horsepower, maxspeed, broadTires, price, interior, luggageCompartment, airbags, antiLockBrakes, catalyticConverter, recyclableMaterials, consumption, leatherSeats, powerWindows, number(1), number(2),…, true, false, spacious, sportiness, prestige, runningCosts, environment, costOfPurchase, security, comfort, family}

$constants_{SCL}$(drtConstant) = { socialEngagement, socialBehaviour,
feelings,
impatience, destructivity,
eyeContact,
hobby,
...}

A-Box: Generic Specification and Example

Before having a look at a concrete instantiation of an A-Box let us first provide a precise definition of the notion of an A-Box.

Pseudo_A-Box	::=	(Type_Intro^{Set}, Attrib_Val_Stat^{Set})
Object	::=	V | C
Type_Intro	::=	Object : T
Complex_object	::=	Object | {$Object_1$,…,$Object_n$} | (Object,Object) | [$Object_1$,…,$Object_n$]
Attr_Val_Stat	::=	attr_name(Object) = Complex_object

In words, a Pseudo_A-Box is a pair consisting of 1. a set of introductions of objects Object:T and 2. a set of attribute-value statements. Each statement specifies the value of a particular attribute of an object (represented by a variable or a constant). A proper A-box (given some T-Box) is a Pseudo_A-box such that all objects that are used in the aforementioned statements are introduced in Type_Intro. Furthermore, all statements have to comply with the T-Box specifications. An A-Box can be represented by means of linear representations, e.g.:

({ v_1:scene,
peedy:person,
robby:person},
{
participants(v_1) = {peedy, robby}

}

)

Example 2

Note that this is a proper A-Box. We do not require that the values for *all* the attributes of an object are specified. Furthermore, the type of a constant is specified in the A-Box although this is not strictly necessary since it is already laid down in the T-Box. Alternatively, this A-Box can be represented as a network as sketched in the previous section.

Further Issues

Representing the Common Ground

In the course of a scene, (dialogue) acts by the participants will lead to updates of the common ground. As we already pointed out, these updated versions of the common ground are not represented in the Scene Descriptions. Rather, the M-NLG keeps track of the common ground (i.e., it updates it) when it generates the form of the acts that are part of the scene. It uses a module internal store for this purpose.

The (initial) common ground is primarily used to determine which objects the participants can refer to and what properties they may use to do so: an object can only be referred to if it is part of the common ground, and only those properties which the object has according to the common ground can be used to *identify* the object.

Note that we are glossing over a number of complications. Firstly, if there are more than two interlocutors the common ground of one pair of interlocutors can diverge from the common ground of another pair of interlocutors. For instance, suppose that there are three interlocutors: A, B and C. In that case we could have three common grounds: one for {A,B}, one for {B,C} and one for{A,C}. To avoid this complication we assume that these three common grounds are equal to each other. Hence, we need to represent only one common ground.

Secondly, we have assumed that whenever an interlocutor introduces some piece of information (e.g., by means of an assertion), it is automatically added to the common ground. This means that we do not deal with situations where interlocutors disagree with each other and therefore do not automatically accept all new information. In order to model such situations we need a more fine-grained representation of the common ground. Consider a scene where there are two participants A and B. In this scene we still use one common ground, but it needs to be carved up into two (possibly overlapping) parts: one part containing the information A is committed to and one part containing the information B is committed to.

Thirdly, there are situations where interlocutors misunderstand each other. In that case, there is no longer a common ground at all. Rather, each interlocutor has its own ideas about what is part of the common ground and these ideas can diverge. In order to model such situations, we would need to assign a separate representation to each interlocutor of what s/he believes to be the common ground.

Finally, we assume that the common ground only contains information which the interlocutors are committed to. This means that we cannot use it to record for instance discussion of objects whose existence none of the interlocutors is committed to. Take the following question: "Does this car have an airbag?" This utterance does not lead to a common ground in which an airbag for the car in question is present, after all the very purpose of this utterance is to find out whether such an airbag exists. However, "It is included for free with this extremely safe car" is a valid response to this question. Here, the 'It' refers to the airbag, i.e., something which was not part of the common ground. In other words, although the common ground can be used to determine for some referring expressions whether they are appropriate, it does not apply to all referring expressions.

Representing the Dialogue History

We have chosen to represent the temporal relations between acts separately from those acts themselves. This allows us to represent in a natural way underspecified temporal orderings on the aforementioned acts. For instance, we can express that some act v_1 is before v_2 and that v_1 is also before v_3 (leaving the ordering amongst v_2 and v_3 underspecified). Note that the use of names to refer to acts is important here. Roughly speaking, we have the statements: before(v_1,v_2) and before(v_1,v_3). Each occurrence of v_1 , v_2 and v_3 here is a name: not the full representation of the object itself. Suppose that this were not the case. That is each occurrence would consist of the act itself (its full representation) instead of its name. In that case, every time one of these acts needs to be changed, we would need to make sure that it is changed at all the places where it occurs. In other words, we would face a rather serious synchronization problem. Our approach avoids this problem: the act itself it represented only once, it can, however, be referred in other places of the Scene Description by using its name (i.e., the of its node/variable).

For the moment we allow for two binary temporal relations: before and simultaneous. For the first version of the eShowRoom demonstrator this will provide sufficient expressive power. However, at some point we might want to express that some set of acts is simultaneous with another act. In order to achieve this we would need to allow for an operator with which complex acts can be composed out of simple ones.

Note that the current implementation in XML of this specification (see NECA D5a) differs slightly from what is proposed here. In particular, in NECA D5a the representation of temporal relations is not as flat as the one proposed here. Two relations are used: sequence and simultaneous, which can be recursively nested. Such representations are more straightforward to process by modules that cannot change the temporal ordering and only add further information to the acts which live inside it (e.g., the speech and animation generators). However, at those processing stages where the temporal ordering is not yet completely fixed, a flat representation which allows for underspecification and incremental extensions might be preferrable.

Non-communicative Acts

Currently, acts other than dialogue acts are neglected in the T-Box (but note that the dialogue acts do cover non-linguistics acts such as gestures, as long as they have a communicative goal). For the eShowRoom domain this seems appropriate. For other domains, it might be necessary to also include non-communicative acts. For instance, we could introduce a subtype agentMovement of the type nonCommunicativeAct. An agentMovement would have a

number of attributes, such as actor, origin, destination and mannerOfMovement . Note that non-communicativeActs can be part of the history of a scene alongside dialogue acts. What is more, the current approach allows us the specify that two acts (e.g., a dialogue and a non-communicative act) occur simultaneously.

Note that some non-communicative acts will not be included in the Scene Description which is produced by the Scene Generator. In particular, pointing acts for identifying objects will be introduced only later by the submodule of the Multimodal Natural Language Generator which produces referential acts.

Semantic Content and Dialogue Act Types

When representing a dialogue act, there is a choice to be made about how much of the meaning of a dialogue act is put into its semantic content attribute and how much is captured by the dialogue act type. A number of consideration seem relevant here:

- To what extent is the semantic content modelled by the Scene Generator? If the Scene Generator does not use deep semantic content it seems artificial to add it later on in the generation process.

- What type of input is more portable to new domains? In principle, semantic representations should not be domain-specific. Of course, different domains might have different predicates, but the structure should remain roughly the same. This suggests that if a Natural Language Generator takes primarily semantic structure as its input, it might be easier to port to new domains. On the other hand, if it generates primarily on the basis of domain-specific dialogue act types, portability might be more difficult to achieve.

Sharing Resources

This document concerns the messages which the SG sends to the M-NLG. Additionally, there is information which the SG and M-NLG might share as a resource. For instance, when they are initialised both of them might require the T-box specification.

Related Work

This paper reports on representations of the content of Scenes that involve (communicative) interaction between two or more characters. Various efforts for developing representation languages are relevant to this work. There is the work in the RAGS project (The RAGS project, 2000) that focuses on defining representations for the purpose of natural language generation of documents. Within this context, network representations (they are called the 'objects and arrows notation') are proposed. In particular, the treatment of rhetorical relations, which have been neglected in this paper, could be relevant to NECA project. A seminar on 'Coordination and Fusion in Multimodal Interaction' convened in October 2001 in Dagstuhl. The slides of 'WG4: Multimodal Meaning Represention' by Laurent Romary address a number of issues that also come up in this paper.[5] In particular, Romary's slides contain four basic constraints

[5] See http://www.dfki.de/~wahlster/Dagstuhl_Multi_Modality/

for meaning representations (the focus is on the meaning of individual utterances): uniformity, incrementality, extensibility and the availability of a clear and explicit semantics. Romary distinguishes meaning representation from domain model representations or ontologies such as OIL and DAML.[6] In this paper, we have seen that the two cannot be entirely separated so easily: the semantic content of dialogue acts is dependent on the concepts which are available in the application domain (i.e., its ontology).

The representations that are proposed in this paper should not be confused with annotation schemes for text or dialogue. Although there are commonalities, the purposes of these representations are quite different: Scene Descriptions are meant for processing by a machine, whereas annotation schemes are tailored for the use by human annotators. Nevertheless, we might be able to make good use of the dialogue act taxonomies that have been proposed as part of annotation schemes.[7]

Neither do Scene Descriptions serve the same purpose as the multimodal presentation mark-up language for affective presentation proposed in Descamps et al (2001). The language proposed there is very close to annotation schemes for text, rather than content. Their representations can be processed almost directly by a player technology, whereas the representations which are proposed here first need to be further processed by the Multimodal Natural Language Generator, the Speech and the Animation Generator.

The most closely related work we encountered is that of De Carolis et al. (2001) which describes their Affective Presentation Markup Language (APML). That document focuses, however, more on the representation of rhetorical relations rather than semantic content. In this document, the emphasis is exactly the other way round.

References

- Bäuerle, R. & E. Zimmermann (1991). Fragesätze. In: Von Stechow, A. & D. Wunderlich (eds.), *Semantik/Semantics: Ein internationals Handbuch der zeitgenössischen Forschung*. Walter de Gruyter, Berlin/New York, 333 – 348.

- Cohen, F. (1929). What is a question? *The monist*, 39, 350 – 364.

- De Carolis, B., C. Pelachaud & I. Poggi (2001). Interactive Information Presentation by an Embodied Animated Agent. In: *Proceedings of the International Workshop Information Presentation and Natural Multimodal Dialogue (IPNMD – 2001)*, Verona, Italy, 14 – 15 December 2001, 19 – 23.

- Descamps, S., H. Prendinger, M. Ishizuka (2001). A Multimodal Presentation Mark-up Language for Enhanced Affective Presentation. In: *Proceedings International Conference on Intelligent Multimedia and Distant Education (ICIMADE – 01)*. Fargo, North Dakota, 2001, 9 – 16.

[6] See http://www.daml.org/

[7] See, for instance, the resource page at http://www.sigdial.org/ for a list of various coding schemes.

- Kamp, H. & U. Reyle (1993). *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, Kluwer Academic Publishers, Dordrecht.

- NECA deliverable D5a: *Initial specification of the description language*.

- Ortony, A., G. Clore & A. Collins (1988). *The Structure of Emotions*. Cambridge University Press. Cambridge MA.

- Partee, B., A. ter Meulen & R. Wall (1990). *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, Dordrecht.

- The RAGS project (2000). The RAGS Reference Manual. University of Brighton/University of Edinburgh. Available at http://www.itri.bton.ac.uk/projects/rags

- Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge.