Research Report on Adaptive Multimodal Fission and Fusion

Authors: Mick Cody, Fred Cummins, Eva Maguire, Erin Panttaja, David Reitter

Project: IST-2001-38685

Project Title: FASiL - Flexible and Adaptive Spoken Language and Multimodal Interfaces

Workpackage: WP5

EC Project Officer: Kimmo Rossi

Keywords: Multimodal Interaction, Fission, Signal Fusion, Adaptability, Natural Language Generation, Wizard-of-Oz Method

Contents

Chapter 1

Executive Summary

In the year since the last Fission/Fusion report, much work has gone into the implementation of a Fission workbench for creating adaptive multimodal user interfaces. The background of this process and recent work in Multimodal Interaction is included in the Research report on Multimodal Fission/Fusion models presented as Deliverable number D 5.1 in May of 2003.

This report will describe the progress we have made on the Fusion services, the fission services and Multimodal Workbench, a project on using Centering theory to add pronouns to our grammars, and an overview of the structure of the MUG grammars for FASiL.

We follow with a discussion of the evaluation of multimodal systems. This includes an overview of the Wizard of Oz experiments in English, Portuguese, and Swedish, as well as a description of an evaluation methodology for mobile multimodal systems and plan for the evaluation of the current MUG grammars.

At the close of the paper are appendices describing the VPA 1.5 integration process, dissemination activities pursued as part of WP5, and a selection of example grammars. The full set of grammars wil be available in deliverable D.5.4 Adaptive multi-modal fission/fusion service.

Much of the text in this report comes from papers which have been published (or will be published) in various conference proceedings. See Chapter 8 for a full listing of references and dissemination activities related to Multimodal Fission and Fusion.

1.1 Multimodality

We attempt here to examine some of the possibilities of the use of coordinated multimodality, in which the system may present screen images and audio which work together to present a view to the user, and in which the user may use voice and gesture together to convey a command to the system. ("Send this [point] to him [point].")

Multimodality is a tool for allowing a user to choose between different ways to access a system. In some systems, multimodality enables access to functionality that would not be available through unimodal means. We, however, are looking specifically as multimodality as a mechanism to allow a user to choose the right modality for himself for a given point in time. As much as possible, all functionality needs to be available in each mode.

Realistically, some features will be more intuitive and easier to use in some modes. For a user who is deaf or hard of hearing, voice messages may be played using an avatar. This is less convenient than text would be. By the same merit, text messages played via voice will use TTS, which is harder to understand than straight text. These are compromises based on the state of the art and the preferred modality of the user.

In addition, interfaces in different modalities need to be different. Some users can use multiple modalities, and we want to offer them that option, in the situations in which they choose to use them, without sacrificing the best possibly functionality and interface in each individual mode.

Some examples of situations in which multimodality is natural:

When a GUI presents a list of options (read, reply, delete), it might be more natural to click. In a VUI-only system, the user may not notice features that are not explicitly offered.

When driving preparing some food in the kitchen, a user might want the ability to look briefly at the GUI for context, and to be reminded of options, but then dictate a response by VUI.

If a user isn't sure of someone's name (eg. John Smith or Joe Smyte) it might

be easier to use GUI to see the match, instead of using trial and error against a grammar that only recognises valid names.

When a user is certain of a name, it will be more natural to say “to Kerry Robinson and Sara Holm” than to brouse through long and possibly slow drop-down menus.

Multimodality is defined, in the context of VPA2 as:

“A single application that MUST accept speech input, touch screen input and keyboard input and MUST respond with speech, text or graphics. All functions available MUST be accessible in speech only or graphics only modes. At each turn, the user SHOULD be able to input one of the following modes: speech, touch screen or keyboard (so not in combination). All users MUST experience the same interface but MAY choose which parts of it to use according to their abilities and preferences. The user experience of the system MUST be equivalent regardless of the input and output modes used.”

That is, VPA2 works with sequential multimodality.

We do this research in the context of delivering mobile solutions to users, and so have chosen to implement our systems for small-screen devices, with voice input, touch screens, and buttons. The theories inherent are applicable to other domains, other systems, and other modalities.

1.2 Adaptivity

We make a distinction between adaptive systems and adaptable ones. Both adaptive and adaptable systems present challenges, as user expectations may be confounded, and interfce consistency needs to be maintained despite variation in surface realization. We see a role for adaptable systems where an *information bottleneck* arises, e.g. because of the use of a small screen device, situational constraints, or changing user preferences. In these cases, we address the problem of adapting the output to the situational demands by generating multiple variants, and selecting among them based on a fitness function which takes these constraints into account. Adaptive systems, on the other hand, change over a longer time scale to match the user’s (or group of users’) needs or skills. Our current methodology

does not involve sufficient testing time to experiment with such adaptivity, though it could be modified to do so.

Fission is an adaptive and adaptable system in that it allows the dialogue manager to specify ways in which the dialogue should adapt to the user. Currently, the dialogue manager can define the level of experience the user has with the system, and the degree of direction present in the dialogue. In addition, by controlling the ordering of the dialogue and the form of confirmation (implicit or explicit), the dialogue manager can adapt to the needs and desires of the user.

1.3 Abbreviations and Acronyms

ASR	Automatic Speech Recognition
FASiL	Flexible and Adaptive Spoken Language
GUI	Graphical User Interface
MUG	Multimodal Unification Grammar
PDA	Personal Digital Assistant
PIM	Personal Information Manager
TTS	text-to-speech
UI	User Interface
VPA	Virtual Personal Assistant
VUI	Voice User Interface
WOz	Wizard of Oz
WOzOS	Wizard of Oz Operating System
WP	Work Package

Chapter 2

Multimodal Functional Unification Grammar

It's a common perception that some meetings are more effective than others. Those meetings that involve the physical presence of participants allow them to rely on multiple communication channels (multimodality), among them natural language, eye gaze and body postures. When channels are missing, such as in a call conference, communicative elements such as topic tracking (coherence) and turn-taking behavior become harder to manage. This is equally true in user interfaces: when restricted to unimodal communication, and this single channel is limited in bandwidth, noisy, or otherwise error-prone, humans encounter difficulties - for example when they use a small-screen computer interface or a voice-based dialogue system over the phone. Humans can usually integrate multimodal information without effort, which leads us the ask: can multimodality improve language-based interaction in bottle-neck devices?

In this chapter, we discuss UI on the Fly, a dynamic output generator for multimodal interfaces that goes beyond the sequential use of input and output methods available in today's human-computer interfaces. UI on the Fly aims to ensure cross-channel coordination for both input and output, so the channels (touchscreen, voice) can be used in parallel. These interfaces convey not just redundant, but also complementary information. For example, they can augment a graphical user interface (GUI) with helpful audio commentary. In mobile situations, screen-based output may be simplified, or eliminated entirely, in reponse to a specific use situation, e.g. when driving. Similarly, the system can adapt to the needs of hard-of-hearing or visually impaired users.

An adaptive multimodal system cannot have a hard-coded interface. While a traditional user interface designer would specify the exact layout and timing as well as the exact wording of graphical and voice-based interface elements, an adaptive interface needs some leeway. However, adaptation needs to be constrained and an algorithm must be defined to choose the right adaptations for the present usage situation, for the particular user and for the device the software currently runs on.

In FASiL, we address the adaptivity of the user interface with a dynamic generator. Multimodal Functional Unification Grammar (MUG) is a unification-based formalism that provides the means to dynamically generate content. That means, the system takes an unambiguous, mode-independent, language-independent (the same representation for Swedish, Portuguese and English) structure and turns it into actual output shown on the screen and played as speech signal via the user's headphones.

The content we generate is coordinated across several communication modes, which currently include natural language and a GUI. It has an inherent mechanism to ensure *cross-modal coherence*, that is, that content presented in each mode uses the same lexical words for discourse entities (the items that it talks about), or, alternatively, pronouns.

The interface can adapt the content presented in each mode to the user's preferences and usage situation. An objective function defines the trade-off between two measures. 1) Predicted cognitive complexity of the output: the system should utter sentences that are easy to understand under the circumstances that the usage situation dictates. 2) Utility: the system should convey as much information as possible in order to speed up the dialogue. This way, the system can select from among several possible output forms generated by the grammar.

UI on the Fly: Generating a Multimodal User Interface

2.1 Abstract

UI on the Fly is a system that dynamically presents coordinated multimodal content through natural language and a small-screen graphical user interface. It adapts to the user's preferences and situation. Multimodal Functional Unification Grammar (MUG) is a unification-based formalism that uses rules to generate content that is coordinated across several communication modes. Faithful variants are scored with a heuristic function.

2.2 Introduction

Multimodal user interfaces are everywhere. The use of a keyboard and mouse on a desktop PC is ubiquitous, if not natural. However, the click-then-type paradigm of common interfaces misses the cross-modal synchronization of timing and meaning that is evident in human-human communication. With coordinated output, novice users could get explanations (redundant content) and experienced users could receive additional (complementary) information, increasing the bandwidth of the interface.

If a user interface is generated on the fly, it can adapt to the situation and special needs of the user as well as to the device.

While users are not necessarily prone to make multimodal inputs Oviatt (1999), they can still integrate complementary output or use redundant output in noisy

situations. Consequently, this paper deals with generating output. We propose a grammar formalism that generalizes decisions about how to deliver content in an adaptable multimodal user interface used in mobile, small-screen devices such as the one employed for the FASiL Virtual Personal Assistant.

2.3 Related Work

Since Bolt's (1980) Put-That-There system introduced cross-modal coordination in multimodal user input, various projects have investigated multimodal input and output methods. Users display a preference for the touchscreen in map-based positioning acts and object selection Oviatt et al. (1997). WIP André et al. (1993) and other systems Feiner & McKeown (1990); Roth & Hefley (1993) generate static multimodal documents. In an interactive user interface, however, layout should remain consistent (Woods & Roth, 1988, perceived stability).

SmartKom Wahlster (2002) is a recent effort that produces a multimodal user interface, using XML/XSLT techniques to render the output. These are deterministic, which makes soft constraints such as usability hard to implement. SUPPLE Gajos & Weld (2004) overcomes this problem in its model of the user and the expected workload for various interfaces, generating a unimodal (graphical) user interface without natural language generation elements. On the integration side, Johnston (1998) presents a unification-based grammar that recasts multimodal signal fusion as a parsing problem.

Our approach employs a non-deterministic grammar to derive variants which are evaluated with a comparatively simple user and situation model according to their utility (information conveyed) and the projected cognitive load imposed on the user. It also removes the requirement inherent in Johnston's system of explicitly defining rules to integrate multimodal information.

In the following, we discuss the grammar formalism used to create output, as well as consistency and adaptation considerations.

2.4 Formalism

In this section, we will explain how the Multimodal Functional Unification Grammar (MUG) allows us to generate content. Our formalism and the associated evaluation algorithm work closely with a dialogue manager. As input, they receive an unambiguous, language- and mode-independent representation of the next dialogue turn.

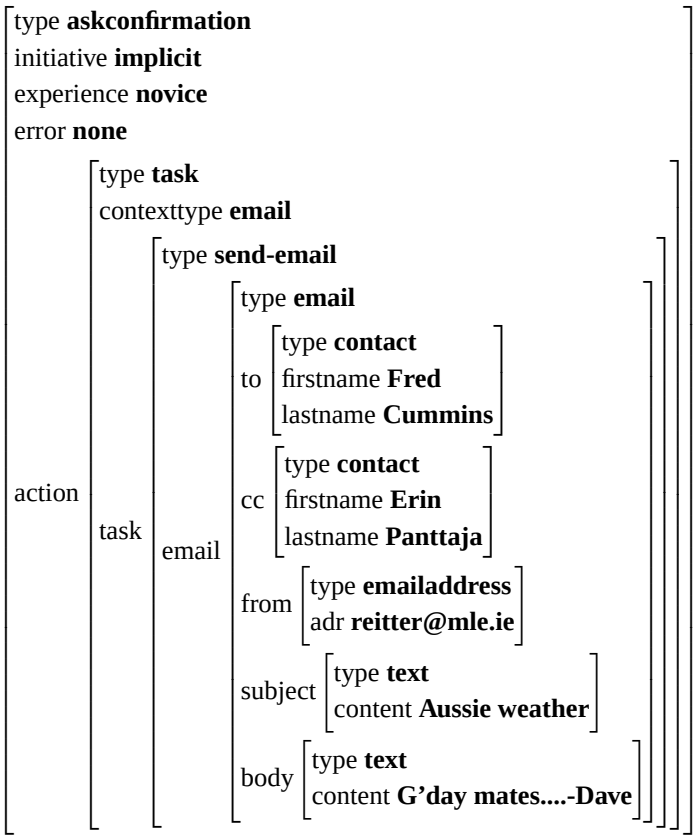
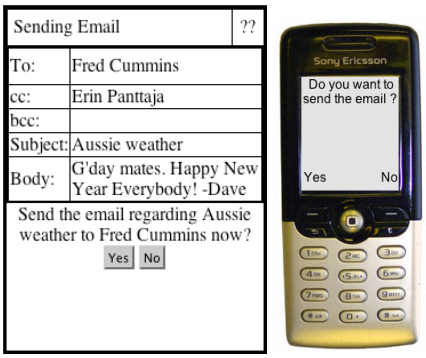


Figure 2.1: Input representation: confirmation of sending of an email

2.4.1 Dialogue acts as input

Although the semantic input is independent of mode (screen, voice) and language (Portuguese), the input semantics are domain-specific. The representation uses the following types of dialogue acts at the top level: ask for missing information, ask for a confirmation of an action or data, inform the user about the state of objects, or give context-dependent help.

An example is shown in Figure 7.3. The input-FD specifies type of act in progress (askconfirmation), and the details of the interaction type. It then specifies the details of the current action, in this case, the email that the user is sending.



(a) (b)

Figure 2.2: a) Voice: “Do you want to send the email? Yes or No?”. b) Voice: “Send the email regarding Aussie weather to Fred Cummins now?”

Furthermore, the dialogue manager may indicate the need to realize a certain portion of an utterance with an attribute *realize*. The input format integrates with principled, object-oriented dialogue managers.

2.4.2 The domain: a personal assistant.

In this example, we have constructed a personal assistant to be used in the domain of sending email messages.

We implemented a MUG for a PDA-size handheld device with a color touch-screen (see Figure 2.2a). The initial steps to adapt it to a mobile phone (Figure 2.2b) involved creating a device profile that uses no GUI widgets and associates a higher cost (see Section 7.4.2) with the screen output, as the screen is smaller. All devices used have server-driven TTS output capabilities.

2.4.3 The grammar

MUG is a collection of *components*. Each of them specifies a realization variant for a given partial semantic or syntactic representation. This representation may be specific to a mode or general. We call these components *functional descriptions* (FDs) in the tradition of the Functional Unification Grammar Kay (1979), from which MUG is derived.

For each output, the MUG identifies an *utterance plan*, consisting of separate constituents in the output. For example, when we ask for missing information ("Who would you like to send the e-mail to?"), the utterance consists of an instruction and an interaction section. Such a plan is defined in a component, as is each more specific generation level down to the choice of GUI widgets or lexicon entries.

MUG is based on the unification of such attribute-value structures. Unification can be seen as a process that augments an FD with additional information. FDs are recursive: a value can be atomic or a nested FD. Values in an FD can be bound to the values in a substructure FD (structure sharing).

To realize a semantic representation R, we unify a suitable grammar component FD with each m-constituent substructure F in R, until all substructures have been expanded. An m-constituent is an FD that has an attribute path $m|$cat, that is, which has been designated as a constituent for mode m. Note that zero or one grammar components for a given mode can be unified with F.

Components from the grammar invoke each other by instantiating the *cat* attribute in the mode-specific part of a substructure. Figure 2.3 shows a component that applies to all modes.

There may be several competing components in the grammar. This creates the ambiguity needed to generate a variety of outputs from the same input. Each output will be faithful to the original input. However, only one variant will be optimally adapted to the given situation, user, and device (see Section 7.4.2). Our final markup is text for the text to speech system as well as HTML to be displayed in a browser, similar to the MATCH system Johnston et al. (2002).

The nested attribute-value structures and unification are powerful principles that allow us to cover a broad range of planning tasks, including syntactic and lexical

choices. The declarative nature of the grammar allows us to easily add new ways to express a given semantic entity. The information that each component has access to is explicitly encapsulated by an FD.

A grammar workbench allows us to debug the generation grammar. We could improve the debugging process with a type-hierarchy, which defines allowed attributes for each type.

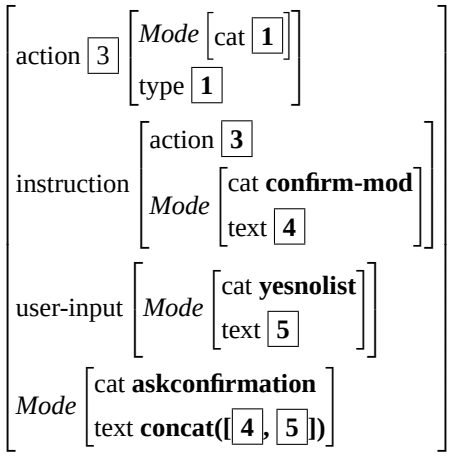


Figure 2.3: A MUG component that handles the confirmation of tasks or user input. The mode in variable *Mode* may be *voice* or *screen*.

2.5 Planning for Coherence

Coherence is a key element in designing a multimodal user interface, where the potential for confusion is increased. Our user interface attempts to be both consistent and coherent. For example, lexical choice does not vary: it is either ‘mobile phone’ or ‘cell phone,’ but it is the same whether it is in text or voice. This is in line with priming effects, which are known to occur in human-human dialogue.

Like humans McNeill (1992); Oviatt et al. (1997), our system aims to be coherent and consistent across all modes. We present redundant content, for example, by choosing the same lexical realizations (never mix *cell phone* and *mobile phone*). We present complementary input in linked components. If, for example, a deictic expression such as *these two e-mails* (by voice) requires the e-mails to be put in focus on the screen, it will set a feature accordingly in the complementary mode.

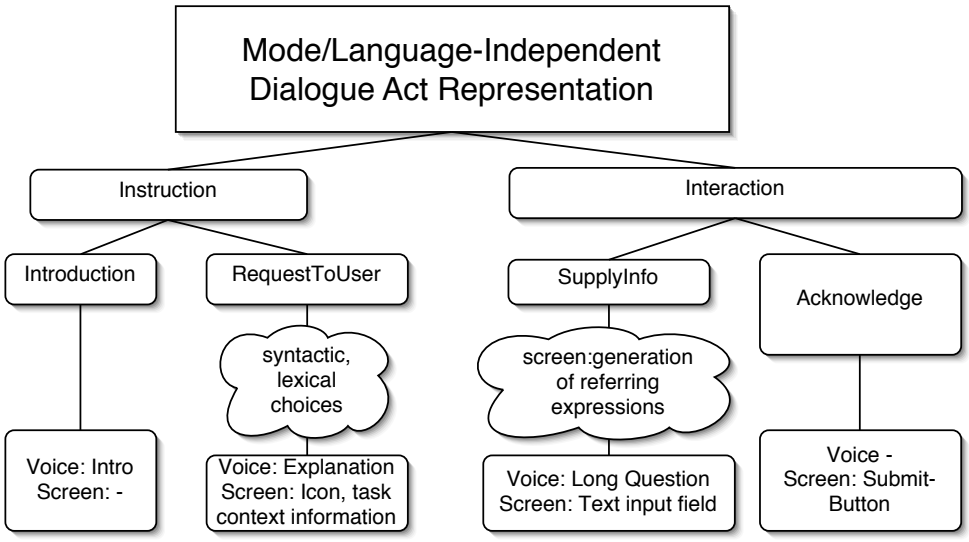


Figure 2.4: Constituents of one generation variant. Voice output: “The system needs additional information from you, before it can send the e-mail. Please specify. Who else would you like to send the e-mail to?”. Screen: “E-Mail, to: M. Cody, ...” (see Figure **??**)

This is possible because of a very simple principle encoded in the generation algorithm: all components realizing one semantic entity must unify. Components may still specify mode-specific information. This is done in a feature named after the mode, so it will not interfere with the realization instructions of a component that realizes the same semantic entity in another mode. The FDs allow us to distinguish information a) that needs to be shared across all output modes, b) that is specific to a particular output mode, or c) that requires collaboration between two modes, such as deictic pronouns. The unification principle replaces explicit integration rules for each coordination scheme, such as the ones used by Johnston (1998), which accounts for the integration of user input.

The nested attribute-value structures and unification are powerful principles that allow us to cover a broad range of planning tasks, including syntactic and lexical choices. The declarative nature of the grammar allows us to easily add new ways to express a given semantic entity.

MUG components can be used to represent a relatively broad range of planning tasks. The utterance plan, a limited number of syntactic choices and lexical choice are implemented in a unified architecture (See Fig. 2.4).

2.6 Adaptively Choosing the Best Variant

The application of the MUG generates several output variants. They may include or exclude pieces of information, which may be of more or less utility to the user. (When information is being confirmed, it should be fully described, but in later interactions, the email could be referred to as 'it.')

For example, several components applied to the sub-FD for *task* in Figure 7.3 may depend more on the screen (Figure 2.2a) or be redundant in screen and voice output (Figure 2.2b). This allows the system to reflect a low benefit for output on the screen if the user is driving a car or to increase the cost of voice output if the user is in a meeting, or reflect the fact that one doesn't hear the voice output on a mobile phone while reading the screen.

The system adapts to the user's abilities, her preferences, and the situation she is in by choosing an appropriate variant. These properties are scalar, and the resulting constraints are to be weighted against each other in our objective function. Each piece of output is scored according to a simple trade-off: a) realize content where requested, b) maximize *utility* to the user, and c) minimize *cognitive load* in perceiving and analyzing the output.

These constraints are formalized in a score that is assigned to each variant ω, given a set of available Modes M, a situation model $< \alpha, \beta >$, a device model ϕ and a utility/time trade-off coefficient λ:

$$s(\omega) = \lambda \sum_{<e,d> \in E(\omega)} u(e,d) + max_{m \in M}(\beta_m t_m(\omega))$$

$$u(e,d) = P(d, \sum_{m \in M} (\phi_m \alpha_m e_{m|realized}), e_{realize})$$

The first part of the sum in s describes the utility benefit. The function E returns a set of semantic entities in e (substructures) and their embedding depths in d. The function P penalizes the non-realization of requested (attribute *realize*) semantic entities, while rewarding the (possibly redundant) realization of an entity. The reward decreases with the embedding depth d of the semantic entity. (Deeper entities give less relevant details by default.)

The cognitive load (second part of the sum) is represented by a prediction of the time $t_m(\omega)$ it would take to interpret the output. This is the utterance output time for text spoken by the text-to-speech system, or an estimated reading time for text on the screen.

Further work will allow us to cover the range of novice to experienced users by relying on natural language phrases versus graphical user interface widgets.

2.7 Conclusion

We have demonstrated a formalism that generates coherent multimodal user interfaces, as well its application in a small-screen email client. As the generation algorithm makes use of both hard constraints and scalar scores, it caters for adaptability. We have proven its functionality and efficiency in a series of examples in the context of a dialogue system, where content is generated in real-time for various usage situations and different devices.

Further evaluation will show whether the fitness function can accurately mirror user satisfaction with a given output variant and whether our form of adaptivity is actually an advantage to users on the go. Without a gold standard for a generation system for dynamic multimodal user interfaces to qualitatively compare against, controlled user trials will allow us to evaluate the usability of the interfaces we have created. Task completion times, user frustration levels, and user satisfaction can then be used to evaluate the success of this model of multimodal interactions.

The underlying formalism is intended to be used in creating, using the MUG Workbench, any multimodal system that can be constructed compositionally, using natural language and other auditory and visual components. As possible examples for future applications, we see a multimodal interface that allows mobile users or users with sensory impairments to traverse information-rich social networks, and a kiosk for multimodal, multilingual access to public transportation options.

2.8 Acknowledgement

The authors would like to thank Stefan Agamanolis, Robert Dale, John Kelleher, Kerry Robinson, and the anonymous reviewers. This research was partially funded by the European Commission under the FASiL project, contract number: IST-2001-38685.

Chapter 3

Multimodal Centering

Multimodal Centering deals with the question of discourse coherence in a multimodal user interface. What is discourse coherence? Using a natural language based interface, a user will talk about many objects in a dialogue with the system, for instance a particular e-mail that is to be sent or a meeting that needs to be planned. Once mentioned, these objects have a prominent status in the discourse. However, they only retain this status for a short period of time. The user may shift his *focus* in the discourse to something new. Similarly, the system can introduce such *discourse entities*, refer to them and eventually shift the focus of the discourse to a new entity.

In many cases, we can refer to these entities in our dialogue with a pronoun (he, she, it). That is not only shorter, but also more natural. When system and users use pronouns and also structure their sentences to make the single steps of the conversation "fit together" (in the eye of an observer), we say the discourse is *coherent*.

Theories of discourse explain why in dialogue users refer to focussed entities with less description – after all, these entities are known from the context: *Can you forward the e-mail from John to Mary, and then delete it.* Most dialogue systems, in turn, do not adhere to this principle: they are overly specific: *Forwarding the e-mail titled Dinner Tonight to Mary O'Sullivan. Deleting the e-mail titled Dinner Tonight.* instead of simply *Forwarded it to Mary, deleted it.* (if there is only one e-mail from John, and only one Mary.) Since the over-specification of referring expressions results in a perceived 'unnatural' and lengthy output, we find it necessary to find a happy medium between maintaining a "not too descriptive" re-

ferring expression whilst avoiding under-specification and the resulting confusion involved.

A computational model of discourse coherence can provide us with an algorithm that determines when to use pronouns, how to refer to discourse entities and how to order our phrases within the sentences uttered. In the multimodal context, it allows us to emphasize the right elements on the screen and control the body posture and gestures of an avatar (which is a usually human-like figure on the screen).

The underlying model of discourse coherence has its roots in Centering theory, which was originally conceived for pure text. We apply it to the multimodal context, with an emphasis on deictic referring expressions (pointing gestures) and natural language dialogue acts.

We show an computational treatment within Multimodal Functional Unification grammar, the grammar formalism developed to generate natural language voice interfaces in combination with natural language and graphical interfaces.

We demonstrate, how the model applies to two practical multimodal generation applications: a virtual personal assistant and, to underline the discourse aspects in an application that could be implemented without the need for the complex architecture of a dialogue system, a language generator that describes people and their social networks.

In Dialogue Systems that rely on Natural Lanaguage Generation as their output it is desireable to maintain a natural communication with the user. Coherence refers to the level to which the shared attentional state of the discourse (during a given segment of utterances) is adhered to and the correct use of referring expressions within that segment.

Centering and Referring Expressions in the Multimodal Context

Abstract

This paper discusses the application of theory of Centering which applies to spoken or written discourse to a Virtual Personal Assistant incorporating multimodal input and output and therefore attempt to extend the theory to cover Multimodal Centering. Multimodal Centering is a theory that allows us to use UI-on-the-Fly techniques to generate context-aware user interfaces which use natural language pronouns and other referring expressions.

3.1 Introduction

Coherence refers to the level to which the shared attentional state of the discourse (during a given segment of utterances) is adhered to and the correct use of referring expressions within that segment. Theories of discourse such as that of Grice (1975) show that in dialogue users refer to focussed entities with less description. Since the overspecification of referring expressions results in a perceived 'unnatural' output it is necessary to find a happy medium between maintaining a "not too descriptive" referring expression whilst avoiding underspecification and the resulting confusion involved.

In dialogue systems that rely on Natural Lanaguage Generation as their output it is desireable to maintain a natural and efficient style of communication with the user. We believe that coherent dialogues go a long way in rendering natural-language based user interaction more natural.

3.2 Centering

One predominant theory of coherence over the last few years has been Centering (Grosz et al., 1995; Walker et al., 1997a). The theory hinges on the following three concepts:

- Linguistic structure groups utterances into discourse segments.
- Attentional Structure provides a representation of the current focus and the focus of the context. It accounts for many choices in the realization of referring expressions in an optimal context.
- Intentional structure consists of discourse segment purposes and the relations between them.

Centering (Grosz et al., 1995) predicts that the perceived coherence of a group of sentences (a segment) involving referring expressions depends on the type of transition from one utterance to the next.

The Attentional structure is defined in terms of *transitions* between the utterances which are defined by linguistic means. The type of transition depends on the focus/attention of the current utterance (U_i), the focus/attention of the previous utterance (U_{i-1}) and what the focus of the current utterance was expected to be following that of the last utterance.

Centering defines a ranking of transitions: some transitions are preferred over others. The rank of the transitions used relates to the perceived coherence of the discourse. Transitions are defined in relation to a small set of data structures (Cb, Cf, Fp) and constraints and rules that operate on them.

3.2.1 Constraints and Rules

Walker et al. (1997a) define the constraints and rules in the theory of centering as follows:

- CONSTRAINTS For each utterance U_i in a discourse segment D consisting of utterances $U_1 \ldots U_m$:

1. There is precisely one backward looking center Cb(U_i, D)
2. Every element of the forward centers list Cf(U_i,D), must be realized in U_i.
3. The center Cb(U_i,D), is the highest-ranking element of Cf(U_{i-1},D) that is realized in U_i.

To elaborate,

1. Cf is the set of FORWARD LOOKING centers which represent entities invoked by an utterance in a discourse segment.
2. Cb is the BACKWARD-LOOKING CENTER and is the highest ranked element in the Cf of the previous utterance that is realized in the current utterance. If there is no previous utterance in the segment (i.e. if this is the first utterance of a new segment of conversation) then there is no Cb.
3. Cp is the PREFERRED CENTER which is an attempt to predict the Cb of the following utterance and is the first member of the Cf of the current utterance. Sometimes the Cp will be what the previous utterance of discourse was about, the Cb, but this is not necessarily the case.

In addition to the data structures Cf, Cb, Cp defined for each utterance, Centering uses a small set of (hard) rules.

- RULES:
  For each U_i in a discourse segment D consisting of utterances $U_1 \ldots U_m$:

1. If some element of Cf(U_i,D) is realized as a pronoun in U_i, then so is Cb(U_i,D).
2. Transition states are ordered. The CONTINUE transition is preferred to the RETAIN transition, which is preferred to the SMOOTH-SHIFT transition, which is preferred to the ROUGH-SHIFT transition.

The type of transition between two utterances U($_i$) and U($_{i-1}$) depends on the following information and table of transitions is shown below.

	Cb(U_i) = Cb(U_{i-1}), or Cb(U_{i-1}) = [?]	Cb(U_i) $\neq$ Cb(U_{i-1})
Cb(U_i) = Cp(U_i)	CONTINUE	SMOOTH-SHIFT
Cb(U_i) $\neq$ Cp(U_i)	RETAIN	ROUGH-SHIFT

3.3 The Generation of Referring Expressions

In order to generate referring expressions automatically it is necessary that the system produces a description of an object which is neither too descriptive or ambiguous within the given context.

Reiter & Dale (2000) provide an incremental algorithm that takes an intended referent and a set of other objects (distractors) and outputs a distinguishing description for the referent - i.e. one that whose referent will be unambiguous in the context. Distractors are “distinct objects answering to the same description”. The attributes selected to modify a referring expression are ordered by “preference” - a relation which is intended to reflect the order in which human writers prefer to describe entities.

3.3.1 Referring Expressions in a Multimodal Environment

The theory of centering as presented in Grosz et al. (1995) and Walker et al. (1997a) operates on utterances in a spoken or written discourse. As it is our intention to show that centering can predict some of the behaviour of referring expressions in a multmodal environemnt it is necessary to investigate how such an environment differs from that of discourse alone.

Salmon-Alt & Romary (2000) note that multimodal interaction differs greatly from discourse in that there are perceptual antecedents for referring expressions. Referring expressions are introduced dynamically on perceptual (visibility on the screen), gestural (introduction of a new entity by the user) or discursive (the mention of a new entity) criteria. In their model, entities are distinguished from one another on the basis of differential criteria.

Our unification grammar produces co-ordinated output in two different modes for output written to the screen and output read using a text-to-speech system through headphones. Since the output is mode specific it is possible to choose different referring expressions (including possibly null realizations) for the two output modes.

3.3.2 Encoding Gesture as a Center

Theories of centering that are applied to discourse alone such as Walker et al. (1997a) must only include in the list of forward-looking centers those referring expressions which were mentioned in the current utterance and the first element of that list from the previous utterance. The visual equivalent of speaking the name of an utterance is by gesticulating towards it and therefore highlighting it for the viewer (we shall assume for the moment that such gestures are always noticed in FASiL). In a multimodal environment which makes no distinction semantically between the two modes which it incorporates any set of forward looking centers would have to include such gestures and rank them according to some refined theory of obliqueness.

In order to encode gesture as a form of referring expression in our application of centering for a multimodal environment we need to investigate how gesture should be encoded with respect to the other centers. Gestures can be used with an accompanying verbal referring expression or as a referring expression on their own. The gesture may occur either before, during or after the spoken referring expression and it may therefore be necessary to decide upon a temporal domain in which a gesture may be bound to a verbal referring expression from the user.

In Johnston & Bangalore (2000) the authors note the following:

> Our approach makes certain simplifying assumptions with respect to temporal constraints. In multigesture utterances the primary function of temporal constraints is to force an order on the gestures.If you say move this here and make two gestures, the first corresponds to this and the second to here. Our multimodal grammars encode order but do not impose explicit temporal constraints. However, general temporal constraints between speech and the first gesture can be enforced before the Feature Structure Analysis is applied.

Our choices with regard to the encoding of gesture must be based on data taken from the transcriptions representing the ways in which people use gesture as a referring expression in a multimodal environment.

3.4 An Analysis of a Multimodal Corpus

In order to decide if centering is a viable theory for prediction of both the binding of user referring expressions in input and the decisions regarding choice of referring expressions in "system" output we investigated the corpus of data collected in a multimodal corpus.

The corpus was collected using a Wizard-of-Oz application (6). The platform permits simulation by an expert of the system's responses and was used to gather empirical data from human subjects about the way in which they would interact with a multimodal VPA such as FASiL. The user remains unaware that the application is being externally controlled.

The centering information for the "system"/user dialogue in the following examples taken from the corpus has been inputted by hand. Note that centers refer to semantic entities and not to the referring expressions which realize them. Therefore when a particular center is referred to by two different referring expressions from one utterance to the next the centers remain the same. Output from the system which does not refer to arguments and which is used purely for communication of status (such as "Checking your Inbox") does not effect the centers. The four types of referring expression which we looked for in the transcriptions were deictics, pronouns and definites.

In the examples taken from the corpus each utterance is follwed by centering information (i.e. the Cb, Cp and Cf as dictated by the constraints in 3.2.1). The centering information given is a discourse analysis of the utterances output by the "system" and user. The analysis of the transcriptions was used in order to provide indications as to the interaction between the modes of gesture and speech. The indices in boxes such as $\boxed{n}$ are used to represent structure sharing of centers from utterance to utterance. The Mode in each example represents whether the user was employing speech and gesture (i.e. the microphone and the screen) or speech alone.

3.4.1 Deictic *this*

The antecedent of a deictic pronoun is either one which has been highlighted or one that has been mentioned in the previous sentence using either a full or deictic reference.

(3.1) Mode: Screen and Voice
U1:I'd like to check my new emails
Cb = [?] , Cf = <[new emails]= [1] >, Cp = [1]
S1:Here are your new emails
Cb = [1] , Cf = < [1] >, Cp = [1]
U2:[gesturing to an email] I'd like to read this email
Cb = [] , Cf = <[email]= [2] >, Cp = [2]

In example 3.1 above U1 has no Cb (as no utterance precedes it in this segment). The system has shown the user their new emails on the screen. The user makes a gesture towards an email which highlights the email. The user then refers to the email with the deictic determiner *this*.

U1:Do I have any email mentioning S's birthday?
Cb = [?] , Cf = < [email mentioning S's bday]= [1] >, Cp = [[1]]
(3.2) S1:Checking your inbox - [System displays one relevant email]
Cb = [?] , Cf = < [a mail mentioning S's bday]= [2] >, Cp = [[2]]
U2:[gesturing to the email] Can you read this please

In example 3.2 above the user gestures to an email and then uses the deictic determiner *this* to refer to that email.

(3.3) Mode: Screen and Voice
S1:Checking your inbox
S2:You have one new email
Cb = [] , Cf = <[one new email]>, Cp = [one new email]
U1:Read this email
Cb = [one new email] , Cf = <[one new email]>, Cp = [one new email]

In example 3.3 the system refers to 'one new email' using a definite expression. The email is the Cb in U1 and the user refers to it using a deictic determiner.

(3.4) Mode: voice
S1:Please repeat search term
Cb = [?] , Cf = < [serch term]> Cp = [search term]
U1:Photos
Cb = [?] , Cf = < [photos]= [2] > Cp = [2]
S2:Checking your inbox.
S3:One email with content 'photos'
Cb = [?] , Cf = < [email with content photos]= [3] >, Cp = [3]
U2:read this email
Cb = [3] , Cf = < [3] >, Cp = [3]

In example 3.4 the system finds and mentions an email which fits the user's request. The user refers to the Cb using a deictic determiner.

So it seems that deictic *this* is employed by the user when the referent has either been highlighted or is the Cb.

3.4.2 Deictic *that*

Deictic that was also used when referring to an object that had been gestured to and highlighted.

(3.5) Mode: voice
S1:Checking your inbox.
Cb = [?] , Cf = <[inbox]= [1] > Cp = [1]
S2:[displays three emails on the screen]
Cb = [] , Cf = <[three emails]= [2] > Cp = [2]
U2:[gesturing to an email on the screen]Read that please.
Cb = [?] , Cf = < > Cp = []

Users seemed to employ the deictic pronoun and determiner *that* instead of *this* when they either felt or wanted to create distance between themselves and an object.

that was used most when in voice-only mode and when there was no visual link to something which was in focus. The user therefore was aware of the object being

in focus yet could not see it.

(3.6) Mode:voice
U1:Can you read the mail from Elizabeth Dixon please?
Cb = [?] , Cf = <[the mail from Elizabeth Dixon]= [1] > Cp = [1]
S1:[System reads out contents of email]
U2:Could you reply to that please?
Cb = [1] , Cf = < [1] > Cp = [1]

In example 3.6 after the system has read out the contents of an email the user refers to the email as *that email*. The mail is open and in focus.

(3.7) Mode: voice
[User has been entering information for a Contact]
S1:Is there anything else?
S2:Yes, could you add a mobile number to that please.
Cb = [] , Cf = < [a mobile phone number]= [1], [that]= [2] > Cp = [1]

In example 3.7 the deictic *that* refers to the contact information which is being added to. The contact is in focus and open.

In dialogue if one locutor used the deictic pronoun *this* then the other refered to the same item as *that* as shown in example 3.8 and example 3.9.

(3.8) Mode: voice
S1:I heard 'your work progress'.
Cb = [?] , Cf = <[your work progress]= [1] > Cp = [1]
S2:Is this correct?
Cb = [1] , Cf = < [1] > Cp = [1]
U1:Could you repeat that?
Cb = [1] , Cf = < [1] > Cp = [1]

(3.9)
S1:is this correct
Cb = [] , Cf = <[this]= [1] > Cp = [1]
U1:yes that's fine please send the email
Cb = [1] , Cf = < [1] > Cp = [1]

The user employed the deictic *that* when they were finished with an object - i.e. *close that, send that*.

(3.10) Mode: voice
S1:Is this correct?
Cb = [] , Cf = <[this]= [1] > Cp = [1]
U1:That's correct.
Cb = [1] , Cf = < [1] > Cp = [1]
U2:Could you send that please?
Cb = [?] , Cf = <[that]= [2] > Cp = [2]

In example 3.11 the system is in voice mode and yet the user still treats it as being in focus or 'open'.

(3.11)
U1:.[dictacting the body of the email]....end email.
Cb = [] , Cf = <[email]= [1] > Cp = [1]
S1:Hang on a second
U2: Send that please. Cb = [1] Cf = < [1] > Cp = [1]

In example 3.12 the user used the phrase "that's everybody". We can assume that the deictic *that* employed refers to the group of people that the email is being sent to whose last member is the CCed recipient - Lisa Stephenson. The group of people that the mail is being sent to realizes Lisa Stephenson and therefore the Cb of U1 in example 3.12 is the Cp of S1.

(3.12)
S1:Cc. Lisa Stephenson, anyone else?
Cb = [1] , Cf = <[Lisa Stephenson]= [1] > Cp = [1]
U1:No that's everybody.
Cb = [1] , Cf = < [1] > Cp = [1]

3.4.3 Pronouns

Pronouns were used by a user when they had either gestured to an object on the screen or when the object had been referred to in the previous utterance (by any

form of expression). Since the “system” did not use pronouns it was not possible to tell whether or not the user employed a pronoun after the system had used one to refer to the same center.

(3.13)
S1:“Appointments today” (displays three apps on the screen)
Cb = [?] , Cf = <[apps for today]= [1] > Cp = [1]
U1:“At 10:30?”
Cb = [2], Cf = <[apps at 10:30 today]= [2] > Cp = [2]
S2:“Elizabeth Dixon”
Cb = [3], Cf = <[app at 10:30 today/app with ED]= [3] > Cp = [3]
U2:“Send her a reminder about this meeting”
Cb = [3], Cf = <[Elizabeth Dixon] = [4] ∈ [3], [3] > Cp = [3]

(3.14) U1:Send an email to ED about the appointment I creted earlier to remind her ab

In example 3.14 above the user employs a pronoun when referring to a person who they have already mentioned in the same sentence.

(3.15) [User is adding information to a contact’s details] U1:Her mobile phone numbe

In example 3.15 above the user is adding information to a contact’s details and the contact’s information is on the screen. The user refers to the contact using a pronoun.

The pronouns employed in examples 3.13 to 3.15 show that a pronoun is used to refer to a center when a definite expression has been used either in the last utterance or in the current utterance. A pronoun can also be used if the referent is open or highlighted on the screen.

3.4.4 Definites

Definites seemed only to be employed when an object is in focus and its type could be used to distinguish the referent from other components or features of an object that is currently in focus.

(3.16)
the body
the address
the subject

The reference *the email* was used when the email was not mentioned in the previous sentence but was in focus due to the task at hand.

(3.17) please send the email

The definite was also used when the user wished to add specificity to something they had said for the system.

(3.18)
U1:check email
S1:checking your inbox
S2:one new email
U2:who is [it] from?
S3:please repeat
U3:who is [the email] from?

The definite produced for an object such as an email was different for that produced by users for contacts. Contacts were referred to by their first name appended onto their last. Users did not refer to contacts as *the/this contact* but used *the/this person* instead.

(3.19)
U1: remind Elizabeth Dixon about this meeting
S1: Checking your contacts
U2: [user gestures to a contact] U3: send this person an email

3.4.5 Conclusions from the Corpus Data

After analysing the mulitmodal corpus we made the following conclusions:

- A pronoun can be used if an object was referred to in the previous or current utterance.

- A definite referring expression should be used if the output is in `voice` mode and the object was not referred to in the last utterance.
- A deictic referring expression can be used by the system in either mode when the referent is distinguishable from type alone (is in focus) and used by the Screen and Voice mode when the referent is highlighted (open or in focus) on the screen.
- Therefore when object is either open or highlighted on the screen it be referred to using either a deictic or pronominal expression.

Analysis of the corpus shows that gesture perceptually highlights an object which can then be referred to using a deictic or pronominal referring expression just as if it had been mentioned in the last utterance. An object which has been gestured to is the most likely candidate for the referent of a pronominal or deictic referring expression and therefore could be considered to be syntactially ranked above any other elements in the current utterance. There are two possible ways of encoding gestures as centers into the grammar which we have developed:

1. Encode a gesture as an utterance in itself which contains one center: the target object which the gesture highlights.
2. Allow any object which has been highlighted with a gesture to automatically become the highest syntactiallly ranked element in the utterance (i.e. the Cp).

3.5 Centering in the Multimodal Unification Grammar

In order to describe our application of centering within a multimodal unification grammar it is necessary to first define the concepts involved in the theory.

- ”object:” A real-world object, represented as a typed sub-structure of a the semantic utterance specification.
- ”semantic entity”: An object in an utterance, with centering information associated. For a single object, there can be only one semantic entity per utterance. In our generation system, we represent an entity as a copy of the attribute value matrix of the associated object, with an additional `centering`

attribute and an `object-id` attribute that holds a unique index to identify the associated object.

- ”center”: Any semantic entity, that has a centering function in the current utterance.
- ”utterance U”: an utterance, represented as a dialogue act specification with several nested semantic entities. U_{i-1} represents the utterance that occurrs before U_i in the discourse.

3.5.1 Semantic Features

Any object in the MUG contains semantic information necessary for the generation of a referring expression for that object. An object contains features which specify its centering status within the current utterance and its centering status within the last utterance. The semantic information stored in an object which is relevant to generating referring expressions is shown in figure 3.20 below.

(3.20)
$$\begin{bmatrix} \text{type } \textbf{Type} \\ \text{objectID } \boxed{1}\ \textbf{ObjectID} \\ \text{png } \textbf{PNG} \\ \text{centering} \begin{bmatrix} \text{current} \begin{bmatrix} \text{cb } \textbf{IsCb} \\ \text{cp } \textbf{IsCp} \end{bmatrix} \\ \text{previous} \begin{bmatrix} \text{cb } \textbf{WasCb} \\ \text{cp } \textbf{WasCp} \end{bmatrix} \\ \text{utterance} \begin{bmatrix} \text{cb } \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- `centering/current/cb` is a boolean feature. 1 indicating that the FD is the current Cb, 0 otherwise.
- `centering/current/cp` is a boolean feature. 1 indicating that the FD is the current Cp, 0 otherwise.
- `centering/previous/cb` is a boolean feature. 1 indicating that the FD was the Cb in the last utterance, 0 otherwise.
- `centering/previous/cp` is a boolean feature. 1 indicating that the FD was the Cp in the last utterance, 0 otherwise.
- `utterance/cb` contains the `objectID` of the object which is the cb in the current utterance.

- `type` holds the type of the object.
- `objectID` holds the number associated with this particular object.
- `png` holds the semantic information about gender and number required to correctly pronominalise a referent.[1]

3.5.2 Allocating the Cf and Cb

From utterance to utterance, the objects in focus shift; hence, their associated centering information changes. The realization of a semantic entity depends on whether a semantic entity associated with the same associated object was a center in the previous utterance. If so, parts of the entities are linked by means of structure sharing.

The constraints set out by four centering transitions (Continue, Retain, Smooth Shift, Rough Shift) can be applied to the unification-based architecture by the links between entities and, additionally, transition constraints.

Semantic entities as a whole are not shared in the dialogue or within the utterance, as their centering information, will be different. Sharing is still possible in the grammar; however, shared structures always share their realizations as well.

1. Centering information is threaded between semantic entities in adjacent utterances which realize the same concept. For every entities $e \in U_i$ and $e' \in U_{i-1}$, for which $e : object - id = e' : object - id$, the following structure-sharing constraint holds: $e : centering : previous = e' : centering : current$.

2. All entities in one utterance that realize the same concept share the same current centering information. For every entity e that is in U_i and that is associated with concept o, $e : centering : current = c_{i,o}$ holds.

3. For every entity e' that is contained in an utterance U_i, $e : centering : utterance$ = u_i. u_i is information specific to an utterance.

   Relations (1) and (2) allow us to define the ranked transitions as FDs that simply unify with the centering structure. Relation (3) ensures that there can be only one backwardlooking center per utterance.

[1] We have not encoded case as a feature as all of our possible argument positions require accusative case.

4. A transition is defined as an FD that holds for all entities in an utterance.

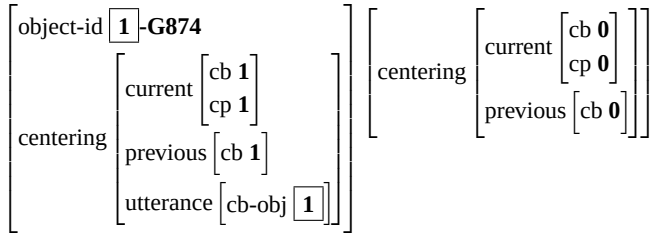
In an utterance U_i, there is some transition:

$$t \in < CONTINUE, RETAIN, SMOOTHSHIFT, ROUGHSHIFT >$$

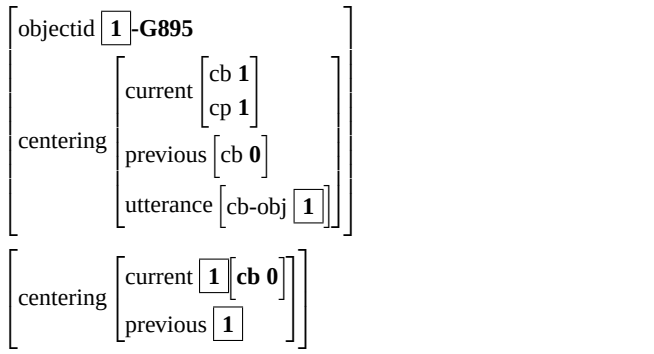
and for each semantic entity must unify with an FD $\in transition(t)$. These FDs are shown in 3.21:

(3.21) The transition function is defined as follows:

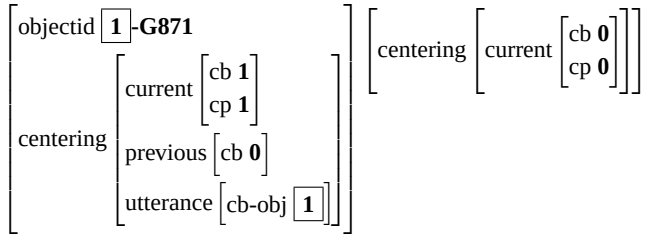
transition(CONTINUE) =



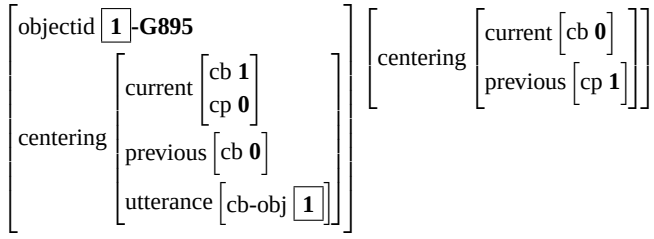
transition(RETAIN) =



transition(SMOOTH-SHIFT) =



transition(ROUGH-SHIFT) =



We define a relation `linkedCenters` which holds if all constraints (1), (2), (3) and (4) hold.

For any group of utterances being input to the MUG the function `linkCenters` is called which instantiates the `objectID`s and centering information of any objects within those utterances relative to the previous utterance. This prolog procedure finds any substructure within the FD representing an utterance which has a `type:` attribute and determines whether or not the object denoted by that object appeared in the previous utterance. If so, the object's values of centering/previous are instantiated as the centering/current values of the previous instantiation of the object. In this way centering information is passed from object to object . The centering information in an object dictates the forms of referring expressions which can be generated for that objects. In order to ensure that only one cb and one cp are possible for each utterance each object is instantiated with a feature `utterance:` which is structure shared between all of the objects within an utterance and which contains the features `cb:` and `cp:`. The value for `utterance/cb` and `utterance/cp` is the `objectID` of the object which has a value of 1 for the `current/cb` and `current/cp`.

3.6 Generation of a Referring Expression for an Object FD

In the MUG any task that instantiates a realization for objects within the grammar must generate referring expressions for those objects. The form of these referring expressions depends on the semantics of the object being referred to which contain information on whether that object was referred to in the last utterance.

Figure 3.1 shows an FD that is to be input into the MUG. The FD in Figure 3.1 represents the information required to instantiate the referring expressions needed to ask confirmation of an action from the user. In this particular instance the task that is being confirmed is that of sending an email. The FD in Figure 3.1 also contains information about the type of action involved and the 'arguments' to that action which include an FD containing information about the email that is to be sent. The task structure being input to the MUG in Figure 3.1 contains object information for the email that is to be sent and the recipients and subject of that email.

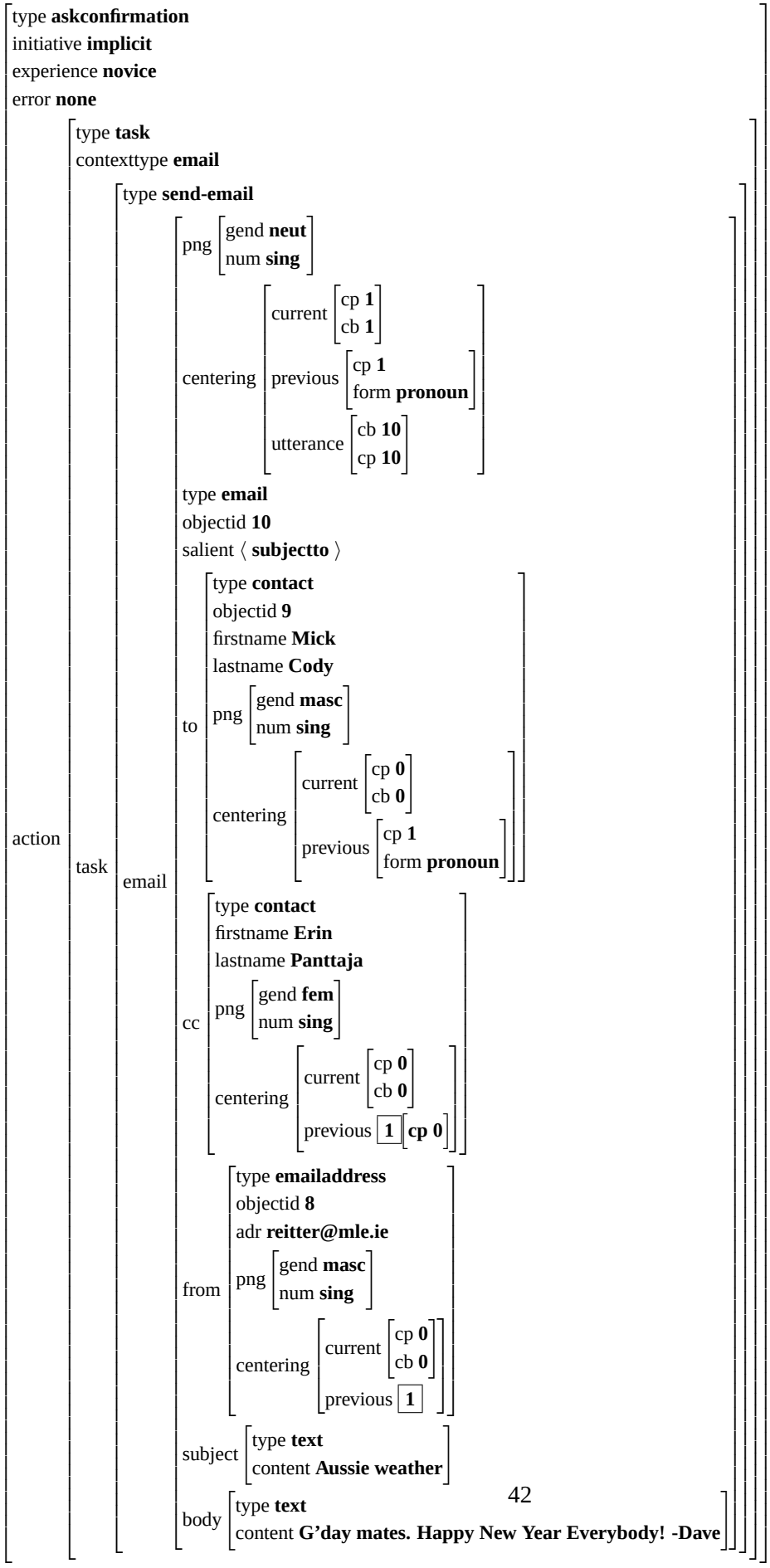
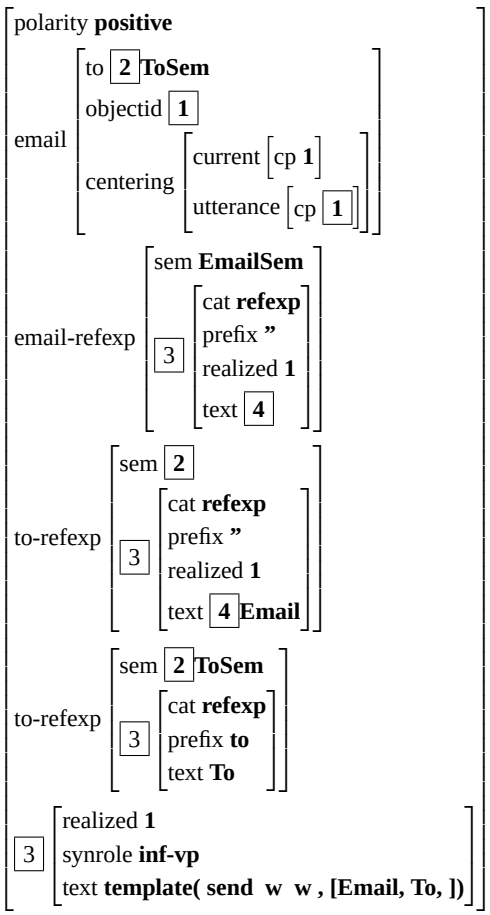


Figure 3.1: Example FD Input to the Unification Grammar

Note that the `png` values for an `email` FD are always instantiated as singular and neutral as this information stays the same regardless of the feature values which the email contains.

The input FD in Figure 3.1 must unify its substructure FD of type `send_email` with the FD in 3.22 below.

(3.22)



The email object is unified with EmailSem and the recipient is unified with the variable ToSem. Each of these structures is passed to the `email-refexp` and `to-refexp` structures respectively which instantiate referring expressions which are structure shared with the realization for the entire FD.

Referring expressions are realized with a set of components of category `refexp`. These components, depending on the centering information provided, decide whether to realize the referring expression as a full, definite, pronominal or deictic referring expression.

Shown in example3.23 is the `email` component which is unified with the email object. The FD structure shares the value within the `subject:` attribute of the input with the `string` attribute and the component `optional_string` instantiates a text value for `Subj` - (" if `Subj` is uninstantiated and the value of `Subj` otherwise). The mode specific `text:` instantiated by `optional_string` is appended onto the text *the email* and the resulting string is instantiated as the mode specific `text:` value of the input FD.

(3.23)
$$\begin{bmatrix} \text{polarity } \textbf{positive} \\ \text{subject} \begin{bmatrix} \text{content } \boxed{1}\ \textbf{Subj} \\ \text{string } \boxed{1} \\ \boxed{2}=\text{Mode} \begin{bmatrix} \text{cat } \textbf{optional-string} \\ \text{prefix } \textbf{regarding} \\ \text{text } \textbf{SubjText} \end{bmatrix} \end{bmatrix} \\ \boxed{2} \begin{bmatrix} \text{realized } \textbf{1} \\ \text{synrole } \textbf{inf-vp} \\ \text{text } \textbf{template(the email w , [Subj])} \end{bmatrix} \end{bmatrix}$$

The examples in example 3.24 show two FDs which may unify with the FD value of `to:`. As shown in example 3.24a a value for `firstname` has been instantiated in an FD then that FD may unify with the structure for `contact_2` below and `text:` is realized as a concatenation of the values for `firstname` and `lastname`.[2] If the address is given the `text` value may also be unified with the value of `adr` as shown in example 3.24b

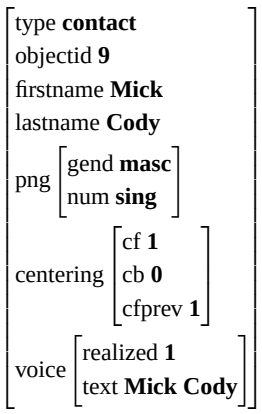
(3.24) a. given(firstname)
$$\begin{bmatrix} \text{firstname } \textbf{Name} \\ \text{lastname } \textbf{Name2} \\ \text{Mode} \begin{bmatrix} \text{realized } \textbf{1} \\ \text{text } \textbf{concat([Name, , Name2])} \end{bmatrix} \end{bmatrix}$$

b. given(adr)
$$\begin{bmatrix} \text{int-adr-is-empty } \textbf{no} \\ \text{adr } \textbf{Adr} \\ \text{Mode} \begin{bmatrix} \text{realized } \textbf{1} \\ \\ \text{text } \textbf{Adr} \end{bmatrix} \end{bmatrix}$$

[2]The component has the name `contact_2` as other components exist which can unify with contact information depending on what values have been instantiated in the input FD.

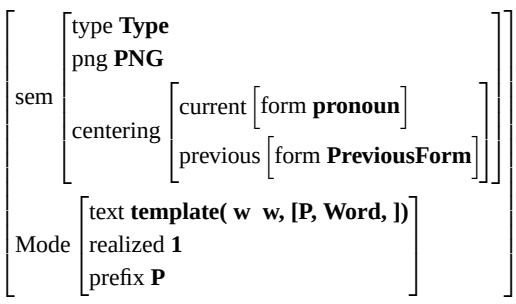
Note that both the FD representing the email and the FD representing the recipient nested within `contact` have a feature `text:`. The value of `text:` is evaluated by the component for a particular `type` of object and represents a realization which may be used as a referring expression in certain circumstances. Note also that the `text:` value is the result of the evalutation of the function `concat`. For simplicity I have filled in the result of this function below which is the concatenation of the values of `firstname:` and `secondname:`.

(3.25)



When the algorithm comes to a `cat:refexp` value in the FD a suitable `refexp` component must be unified with the FD from which `refexp` is called. The `refexp` component instantiates the `text:` variable in the FD which calls it depending on the semantic information it receives from the `png` and `centering:` values of that FD and the value of the `prefix:` attribute which is passed to it. The FD in example 3.25 is unifiable with the FD in the following instance of a `refexp` component.

(3.26)



The following constraints must also hold:

```
/* first pronominalization rule
   at least one of the following conditions must hold:
    - the currently pronominalized concept did not appear in U(i-1),
```

```
    - the CB of the current utterance is realized as pronoun
    This excludes the case that is forbidden by the rule:
    - this element occurred in Cf (u-1), but CB(U) is not realized as pronoun
        */
        ( WasInCF = 0 ;    FormOfCurrentCB = pronoun  ),

/* it MUST have been realized somehow in the previous utterance */

        % not null
        ( PreviousForm=pronoun; PreviousForm=full; PreviousForm=deictic ),

/* cheap lexicon access */
        lex(PNG, Word).

lex(PNG, 'it')  :- PNG === [gend: neut, num: sing].
lex(PNG, 'her') :- PNG === [gend: fem, num: sing].
lex(PNG, 'him') :- PNG === [gend: masc, num: sing].
lex(PNG, 'them'):- PNG === [num: plural].
```

The component in example 3.7 will unify the `png` information from the recipient object. `Mode:` unifies with `voice` and the `text:` value for the `voice` mode is instantiated as 'him'. If we return to the output of the `send_email` component in example 3.22 we can see that the `text:` value instantiated by the call to `refexp` for the recipient information in `to:` is used to refer to the recipient in the `text:`. The `realized:` boolean value is instantiated as 1 if the FD which contains it has been output in that particular mode. A referring expression for the FD representing the email will be instantiated in `text:` through the same process as that for the FD representing the recipient information value and the output of `send_email` for the `voice` mode (depending on the referring expression instantiated in the email) will be something like the simplified segment of the resultant FD shown in example 3.27 below:

(3.27)
$$\begin{bmatrix} \text{voice} \begin{bmatrix} \text{realized } \mathbf{1} \\ \text{text } \mathbf{template(send\ \ w\ \ w, [the\ email, to\ him])} \end{bmatrix} \end{bmatrix}$$

The final function called is `template` which inserts the list values in its second argument consecutively into the `~w` variable values in its first. The `text:` value is then instantiated as the FD in example 3.28:

(3.28)
$$\begin{bmatrix} \text{voice} \begin{bmatrix} \text{realized } \mathbf{1} \\ \text{text } \mathbf{send\ the\ email\ to\ him} \end{bmatrix} \end{bmatrix}$$

3.7 The `refexp` Components

Each `refexp` component has an attribute `sem:` into which the FD for an object is passed when the component is being called by `cat`. Depending on the information contained within `png` and `centering` the component instantiates a string into the `text:` attribute. The `refexp` components are divided into four main categories: full referring expression, definite referring expression, pronominal referring expression and deictic referring expression.

(3.29)
$$\begin{bmatrix} \text{sem} & \begin{bmatrix} \text{type } \boxed{1}\textbf{-G1591} \\ \text{png}\left[\text{num } \textbf{sing}\right] \\ \text{centering} & \begin{bmatrix} \text{current}\left[\text{form } \textbf{full}\right] \\ \text{previous}\begin{bmatrix}\text{cp } \textbf{0} \\ \text{form } \textbf{null}\end{bmatrix}\end{bmatrix} \\ \boxed{2}\text{=-G1662} & \begin{bmatrix}\text{cat } \boxed{1} \\ \text{realized } \textbf{1} \\ \text{text } \textbf{-G1657}\end{bmatrix} \end{bmatrix} \\ \boxed{2} & \begin{bmatrix}\text{text } \textbf{concat([-G2095, , -G2101])} \\ \text{prefix } \textbf{-G1674} \\ \text{realized } \textbf{1}\end{bmatrix} \end{bmatrix}$$

As shown in the FD in example 3.30 a definite referring expression may be used if the object was not referred to by a pronoun in the last utterane.

(3.30)
$$\begin{bmatrix} \text{sem} & \begin{bmatrix} \text{type } \boxed{1}\textbf{ObjectType} \\ \text{png}\left[\text{num } \textbf{sing}\right] \\ \text{centering} & \begin{bmatrix} \text{current}\left[\text{form } \textbf{definite}\right] \\ \text{previous}\begin{bmatrix}\text{cp } \textbf{0} \\ \text{form } \textbf{PreviousForm}\end{bmatrix}\end{bmatrix} \\ \boxed{2}\text{=Mode}\left[\text{cat } \textbf{template-mod}\right] \\ \text{template } \textbf{type} \\ \text{otype } \boxed{1} \\ \text{text } \textbf{TypeText} \end{bmatrix} \\ \boxed{2} & \begin{bmatrix}\text{text } \textbf{template(w the w, [P, ObjectType])} \\ \text{prefix } \textbf{P} \\ \text{realized } \textbf{1}\end{bmatrix} \end{bmatrix}$$

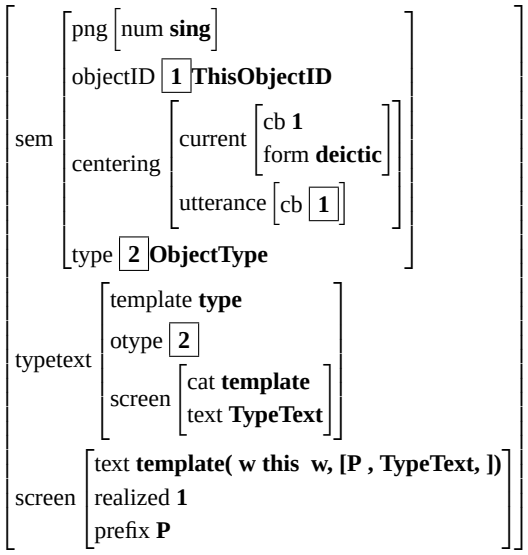
The following constraints must also hold: (PreviousForm=null; Previous-

Form=full; PreviousForm=deictic),
(ObjectType’’).

shows the pronominal referring expressions. If an object was referred to in the last utterance it may be pronominalised according to its semantics. If an object was a pronoun in the last utterance it is excluded from begin referred to by any referring expression other than a pronoun.

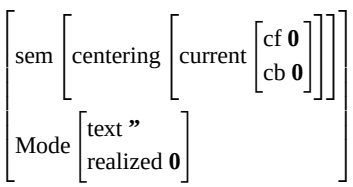
As shown in example 3.31 a deictic expression can be used in screen mode only and when the object is the current cb. This component will also instantiate as the cb any object which is referred to using a deictic expression.

(3.31)



Should an object be empty then an empty text value may be instantiated for that object’s referring expression and realized is set to 0 so that the scoring algorithm does not count the referring expression as output.

(3.32)



3.8 Generating Personal Descriptions from a Social Network Database

In the previous section of this chapter, we have showed how referring expressions are instantiated in a single utterance. A crucial part is still missing: how is a discourse generated? In Section 5, we introduced the `linkCenters` relation. In this section, we will show how it can be applied to a text plan generated with MUG.

In this section, we discuss and demonstrate the generation of anaphoric pronouns in the context of a different grammar that is unrelated to the domain explored in FASiL's Virtual Personal Assistant. The system in question generates brief personal descriptions from databases that contain information about people, including relational information to their business or social contacts. These networks have become increasingly common; examples are Friendster (www.friendster.com) or Orkut (www.orkut.com). The personal profiles stored in these public public databases contain information about a person's name, likes and dislikes and links to their friends.

The original semantic input to the social network grammar is similar. However, since several sentences are generated (which are clearly different utterances in the Centering sense), we need to first come up with a text plan that specifies the utterance semantics and the order of the utterances. Just like during the later text realization step, we use a Multimodal Functional Unification Grammar for this job.

As a second step, all necessary centering-related attributes are instantiated in the utterance semantics, so the linkedCenters relation holds.

In a third step, the markup or texts for each utterance is generated, similarly to the way we did it in the previous section.

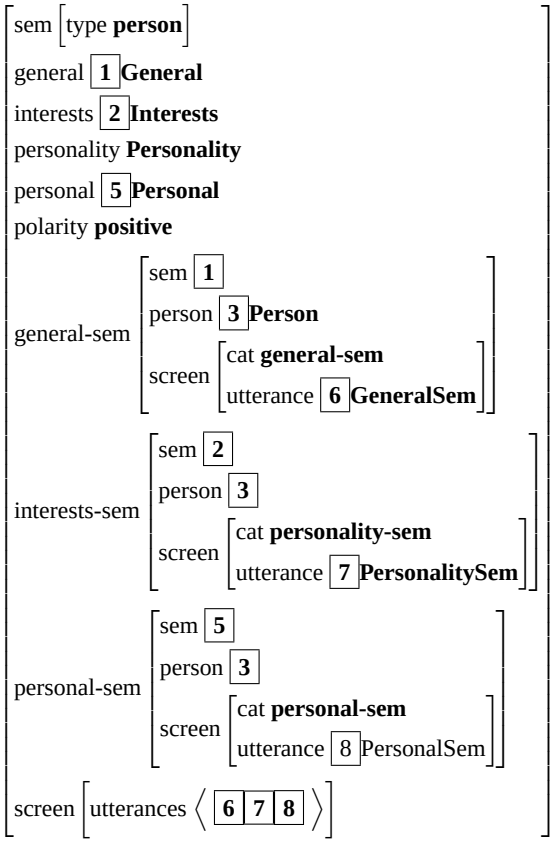
3.9 Grammar components

The information from the profile is instantiated in an FD with the structure shown in example 3.33 below:

(3.33)
$$\begin{bmatrix}
\text{sem} & \begin{bmatrix} \text{png} & \begin{bmatrix} \text{gend} & \textbf{fem} \\ \text{num} & \textbf{sing} \end{bmatrix} \\ \text{firstname} & \textbf{Sarah} \\ \text{lastname} & \textbf{Jones} \end{bmatrix} \\
\text{general} & \begin{bmatrix} \text{describe} & \textbf{I am great fun to be around} \\ \text{rs} & \textbf{single} \\ \text{country} & \textbf{Ireland} \\ \text{hometown} & \textbf{London} \end{bmatrix} \\
\text{interests} & \begin{bmatrix} \text{passions} & \textbf{goingout} \\ \text{sports} & \textbf{rollerblading} \\ \text{activities} & \textbf{chess} \\ \text{favourite} & \begin{bmatrix} \text{books} & \textbf{Faust} \\ \text{music} & \textbf{''} \\ \text{tv} & \textbf{''} \\ \text{movies} & \textbf{''} \\ \text{cuisines} & \textbf{''} \end{bmatrix} \end{bmatrix} \\
\text{personality} & \begin{bmatrix} \text{politics} & \textbf{''} \\ \text{soh} & \textbf{quirky} \\ \text{fashion} & \textbf{urban} \\ \text{smoking} & \textbf{heavy} \\ \text{drinking} & \textbf{socially} \end{bmatrix} \\
\text{personal} & \begin{bmatrix} \text{first} & \textbf{''} \\ \text{best} & \textbf{''} \\ \text{date} & \textbf{''} \\ \text{five} & \textbf{''} \\ \text{match} & \textbf{''} \end{bmatrix}
\end{bmatrix}$$

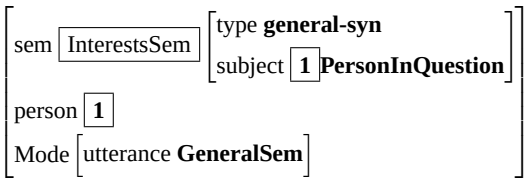
A component called `generate_utterances` passes each substructure contained in `general`, interests:, personality: and `personal:` to the following components:

(3.34)

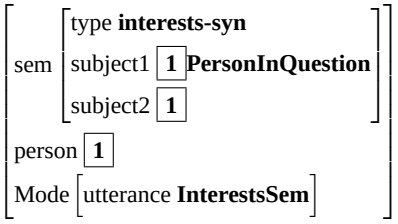


`general_sem` instantiates the semantic information for a subject

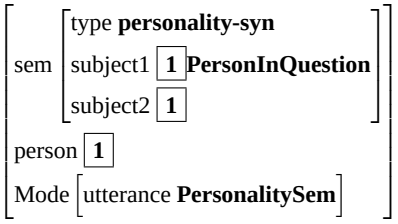
(3.35)

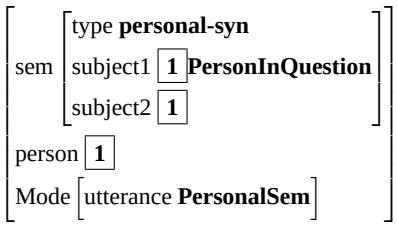


(3.36)

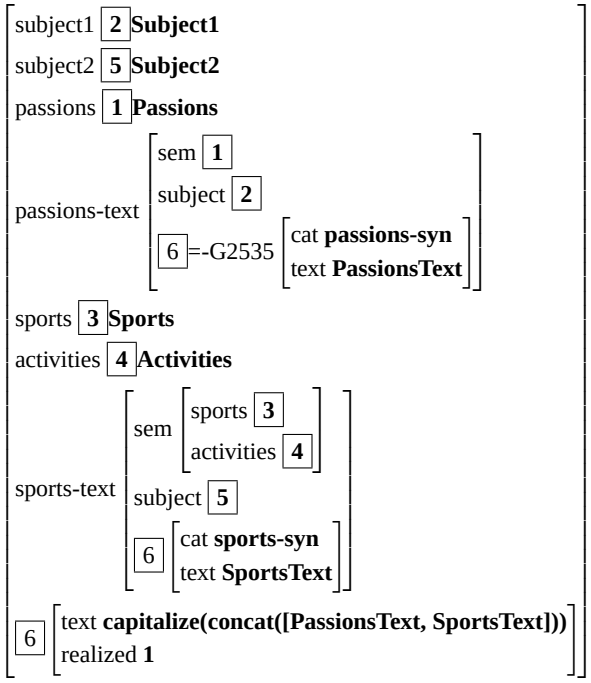


(3.37)



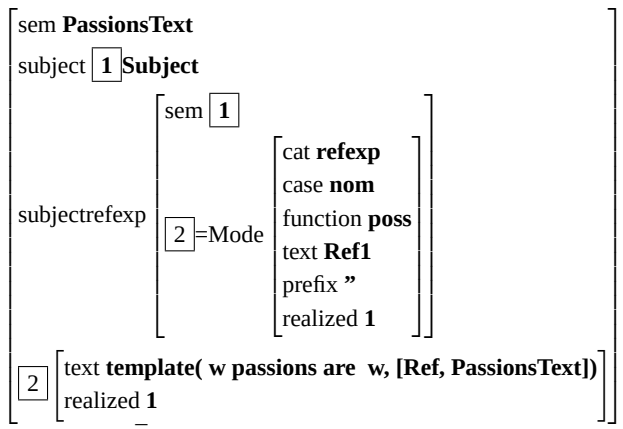
(3.38)


The syntactic information in the output is constrained by unification with the following FDs. Note that each FD which unifies a realization with the `text` value has a corresponding 'empty' component which will unify with an FD with underspecified information.

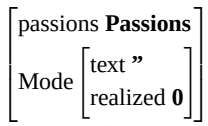
(3.39)


`passions_syn`

(3.40)



PassionsText ”.



Passions=”.

Further components realize other existing information about the person in question.

3.9.1 Examples

The experimental generation grammar, together with the algorithm to generate referring expressions, is able to produce short discourses like these:

(3.41) Sarah is from London, but living in Ireland now . Her passions include going out , and she also enjoys rollerblading as well as chess.

3.9.2 Further Work

Further work is required to insure the right relation of referring expressions within a single utterance. Right now, semantic entities that refer to the same object are still unified:

(3.42) Sarah‘s passions include going out , and Sarah also enjoys rollerblading as well as chess.

Coordinated sentences may be treated as separate utterances. In this case, the text planning grammar did not do so and unified both instantiations of the referring expression (*Sarah*).

3.10 Conclusion

In this chapter, we described a unification-based approach to discourse coherence, in particular with respect to the generation of referring expressions. Far away from a comprehensive theory, we were able to show first investigations into a model of multimodal discourse coherence in the Centering framework.

Chapter 4

Grammars

Grammars for the MUG system are written in a prolog-like grammar formalism. Each component describes a method of display for a given piece of information. This may involve calling for the application of other components. There may be multiple components of a given type; in this case, each applicable component will be unified with the input in turn. Each that successfully unifies will create a variant of the output.

Components may be independent of mode (for example, a user's name may be "George Jones" regardless of the modality being used), or specific to a given mode (radio buttons may be used only on the screen).

4.1 Unification

A component is represented by a Functional Description (FD). Each FD is a collection of attribute-value pairs. A value may be a terminal ("Send email"), a function (concat("George ", "Jones")), or a nested FD.

Unification is a process in Prolog in which two structures are merged into one if the appropriate structures matched.

Here we have a structure describing George Jones.

$$\text{FD} === \begin{bmatrix} \text{firstname } \mathbf{George} \\ \text{lastname } \mathbf{Jones} \end{bmatrix}$$

And a structure showing how to display a name.

FD ===

$$\begin{bmatrix} \text{firstname } \boxed{\textbf{?}} \\ \text{lastname } \boxed{\textbf{?}} \\ \text{text } \textbf{concat(firstname, lastname)} \end{bmatrix}$$

Unifying these two structures, we get:

FD ===

$$\begin{bmatrix} \text{firstname } \textbf{George} \\ \text{lastname } \textbf{Jones} \\ \text{text } \textbf{George Jones} \end{bmatrix}$$

4.2 Anatomy of a component

Here is a MUG component that handles the confirmation of tasks or user input.

component(askconfirmation, _, Mode, FD, askconfirmation_1) :-

FD ===

$$\begin{bmatrix} \text{action } \boxed{3} \begin{bmatrix} \textit{Mode} \begin{bmatrix} \text{cat } \boxed{\textbf{1}} \end{bmatrix} \\ \text{type } \boxed{\textbf{1}} \end{bmatrix} \\ \text{instruction } \begin{bmatrix} \text{action } \boxed{\textbf{3}} \\ \textit{Mode} \begin{bmatrix} \text{cat } \textbf{confirm-mod} \\ \text{text } \boxed{\textbf{4}} \end{bmatrix} \end{bmatrix} \\ \text{user-input } \begin{bmatrix} \textit{Mode} \begin{bmatrix} \text{cat } \textbf{yesnolist} \\ \text{text } \boxed{\textbf{5}} \end{bmatrix} \end{bmatrix} \\ \textit{Mode} \begin{bmatrix} \text{cat } \textbf{askconfirmation} \\ \text{text } \textbf{concat([} \boxed{\textbf{4}} \textbf{, } \boxed{\textbf{5}} \textbf{])} \end{bmatrix} \end{bmatrix}$$

This is a component of category “askconfirmation.” That is, it asks for a confirmation from the user of a given piece of information.

The second parameter (’_’) represents the language. In this case, the component is language independent, as there are no actual words presented, and the ordering of pieces is pragmatic, rather than grammar-based.

The third is the available mode. This component may be used in any mode.

The fourth is a variable to hold the FD.

The final parameter is a name of this specific component. There may be several 'askconfirmation' components, but this is the unique name of the component.

This component calls (via the nested FD with category (cat) confirm-mod) another component that will generate text confirming a task with the user. (for example, "Do you want to send email to George Jones?")

The component then calls a component "yesnolist." Depending on the mode, that may generate a response of the form "Please say yes or no," or a pair of labelled buttons. For an expert user, it may generate nothing at all. On a telephone, it may remap the two input buttons.

4.3 Realized

Another important aspect of the components is the ability of the interaction designer to require the instantiation of certain pieces of information. Certain elements may be marked with an attribute "realize." This is an instruction from the dialoge manager that this piece of information must be present in the output presented to the user.

"Realized" is an attribute set by the MUG engine to indicate that a given piece of information has been faithfully represented in the output, in at least one mode.

Together these ensure that required information is presented to the user.

4.4 Grammars

A grammar may be arbitrarily complicated. It could have a single component for each possible state of the dialoge manager, or it could do natural language generation to put together a response word-by-word and element by element.

The MUG formalism is designed to allow a combination of the two. When a new

mode or language is added, some new components will be required (to represent the input options available, or to create prompts in the new language), but a large number of the components in the existing system would be usable, allowing for faster addition of new modes, and for consistency in interface between modes of interaction.

For more details on the grammar, see Appendix B.

Chapter 5

MUG Workbench – A development environment for Multimodal Functional Unification Grammar

The MUG approach to the design of adaptive interfaces described in this report redefines the role of the designer. Instead of coming up with a complete user experience, designers create a number of (alternative) realizations for smaller communicative functions – be it a part of a sentence in natural language, or an arrangement of “yes” and “no” buttons in a graphical user interface.

However, such specifications (grammars) tend to become hard to extend and debug. The MUG system represents a new tool set to address this issue.

The particular formalism supported is Multimodal Functional Unification Grammar, a grammar formalism that supports several coordinated modes, such as voice prompts or structural and/or language-based screen displays. For each input description, the grammar can generate a range of coherent realization variants, which are ranked by a scoring function in order to optimize the output towards situational and device-related factors.

The MUG System (FASiL Deliverable D5.4) is a development tool that consists of a Formalism, a central algorithm and the Workbench development environment that is the focus of this chapter.

The MUG Workbench works as an inspection tool, which runs a test case, generating all possible *variants* of the output, and then gives access to the various steps

taken during content realization.

The workbench offers several views of the generation process that allow the grammar developers to inspect the details of a single potential output, as a combination of voice, screen or other modes. Developers can run unit tests defined with the grammar or make free inputs, which, in a complete dialogue system, would come from a dialogue manager.

A Development Environment for Multimodal Functional Unification Generation Grammars

5.1 Introduction

When grammar-based techniques for natural language generation (and analysis alike) find their way into collaborative projects or actual application, big grammars tend to become hard to extend and debug. The MUG system represents a new tool set with a graphical debugging environment for functional unification grammars, which is designed to help grammar developers inspect the results of their work.

The particular formalism supported is Multimodal Functional Unification Grammar (MUG, Reitter et al. (2004)), which is similar to Functional Unification Grammars (FUG: Kay (1979), Elhadad & Robin (1992)), but supports several coordinated modes, such as voice prompts or structural and/or language-based screen displays. For each input description, the grammar can generate a range of coherent realization variants, which are ranked by a scoring function in order to optimize the output towards situational and device-related factors.

5.2 Development with MUG

5.2.1 System overview

The MUG System is a development tool that consists of several components. The *MUG Formalism* is a grammar specification syntax. The *MUG Engine* handles the

generation and adaptation process and offers interfaces to connect external components. *MUG Workbench* is a graphical development environment. The workbench works as an inspection tool, which runs a test case, generating all possible *variants* of the output, and then gives access to the various steps taken during content realization. We found this a faster method than the step-by-step execution in a graphical debugger.[1]

5.2.2 MUG Formalism

The MUG formalism is close to Prolog syntax. One or more grammar files contain components, which are usually made up of one big FD (additional disjunctive or conjunctive unifications are allowed). Variables can be named and always start with an upper-case letter. The grammar writer is allowed to add detailed comments about components or their parts. MUG rules (*components*) are attribute-value matrices (AVM), which regularly have internally shared substructures propagating information between the different levels of linguistic representation. The grammar formalism implements this functionality by using named variables.

Unification-based grammars just like other object-oriented formalisms need to balance off the safety of strongly typed classes, which give error messages at an early stage, and the fact that no typing supports exploratory programming and quick prototyping. We address the issue with a type hierarchy that is used to issue non-fatal warnings in case of possibly ill-formed substructures in dialogue act input and grammar rules.

5.2.3 Inspecting variants of output

In the *variants view* (Fig. 5.1), these variants may be inspected and compared: the workbench lists, for each variant, the components used. Variants with the same generated text, but different structures, are automatically flagged, as we found this to be a common problem during the development of grammars. Generally, the variants view is a good way to deal with faulty or extraneous variants.

Misspelled variable names, but also variables in the wrong positions in AVMs

[1] The generation system including the workbench will be available shortly as open source, with documentation and for free.

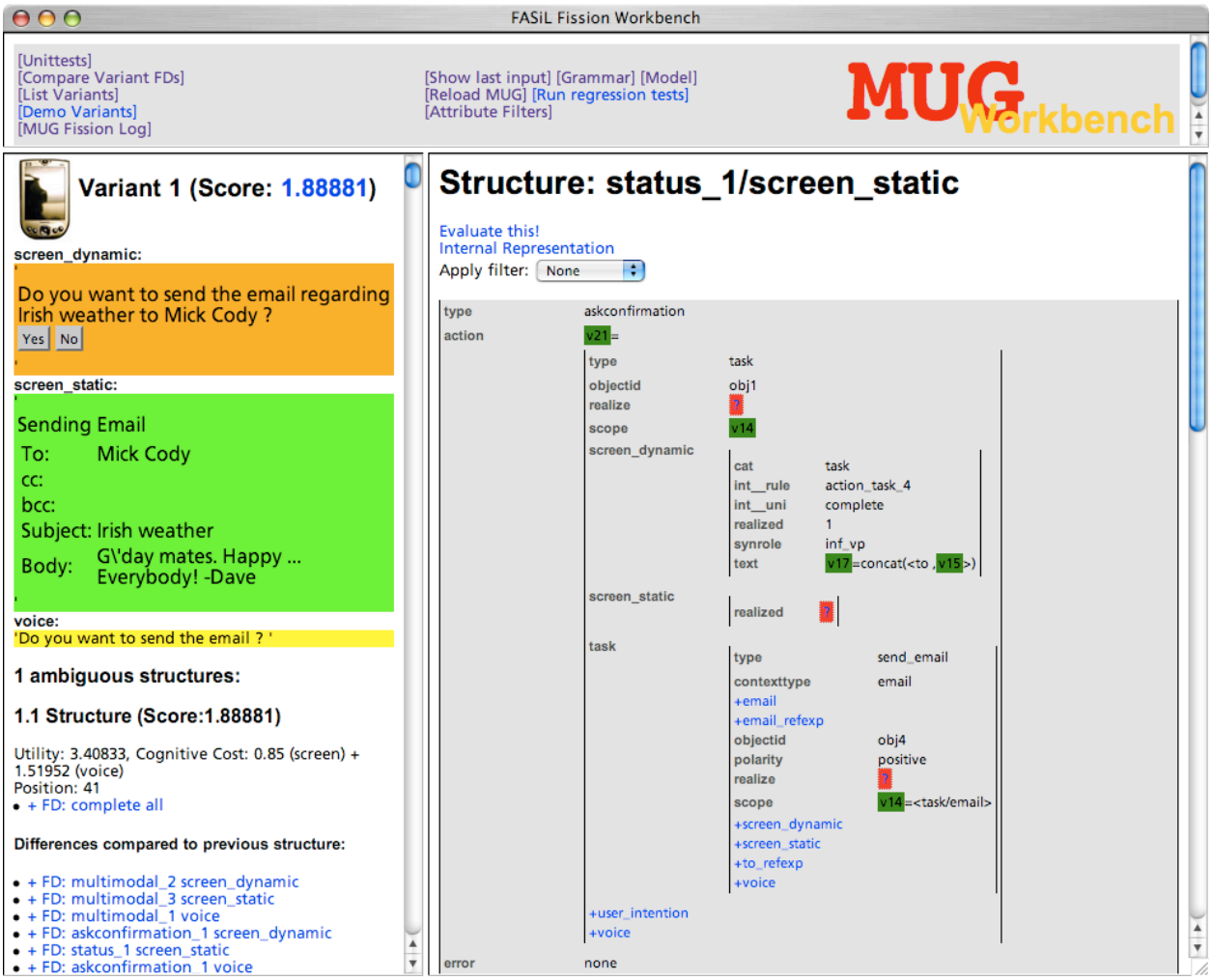


Figure 5.1: Variants and a large AVM. Attributes can be collapsed.

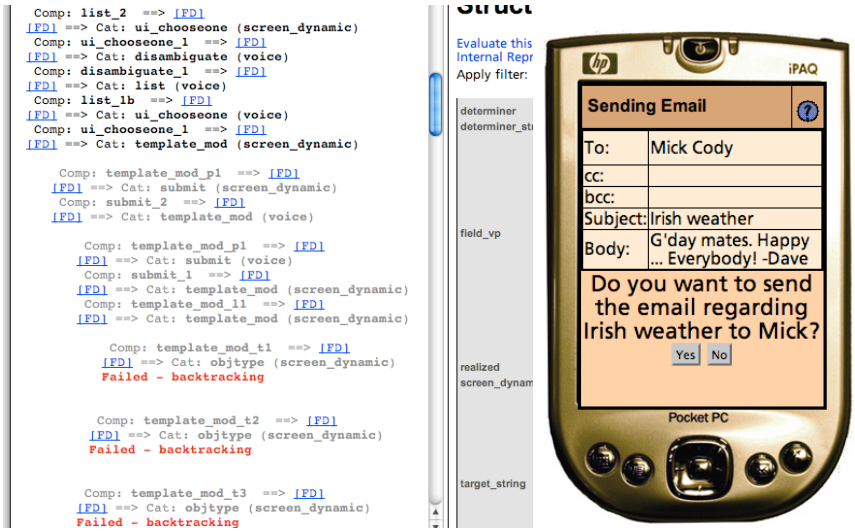
are a common source of errors. Such variables remain unbound. The workbench marks them clearly in the display. The developer can also inspect variables easily and collapse or filter the rather large feature structures. Syntax errors are shown, when the grammar is loaded via the workbench user interface.

5.2.4 Log view

This view of the process is purely logical: there is no conceptual time-line in unification-based grammars as in procedural programs. We found that a more procedural view may help to spot problems with variants that failed to come up, furthermore it is a way to spot efficiency bottlenecks or to simply learn about how

the formalism works. We offer a *log view* (Fig. 5.2) that enumerates all the steps that the MUG interpreter takes to apply a grammar to the input. These steps are shown for each variant of the output. This view allows inspection of the state of the sub-substructures as they were before and after a component (for a given mode) was applied.

A useful feature in this view is the marking of steps that were undone by means of backtracking, because – at a later stage in the generation process – an application of a rule failed. In many cases, the cause of the failure is a bug in the grammar. In other cases, it is desired behavior, but computationally inefficient. Such effects are visualized in the log view.



(a) (b)

Figure 5.2: a) In the log view, steps taken back during backtracking (because they didn't lead to valid solutions) are greyed out. b) life-size results can be demonstrated from the workbench (design: E. Panttaja)

5.3 Applications

The MUG Workbench aided two experienced and one novice grammar writers to create a multimodal UI for personal information management (handling e-mail), which contains 126 components (190 with disjunction compiled), and a second, smaller MUG (39 comp.), which generates a short coherent discourse with pronouns. To test (see Panttaja et al. (to appear 2004)) and also demonstrate the

grammar, multimodal output can be made on any networked device (e.g. PDA) with an HTML client, with text-to-speech voice rendered on the server, as well as on simulated devices on the local workstation. A dialogue system in Java will demonstrate the use of MUG a complete environment.

Chapter 6

A Platform for Multimodal Wizard of Oz User Interaction Studies

In order to collect Multimodal Interaction data in a similar Personal Information Manager setting a Wiazard of Oz study was constructed. A platform to do this was created and following the study itself, which was conducted January-March 2004, further enhancements were made to this platform to prepare for general release. This chapter outlines the operation and use of the platform.

A Platform for Multimodal Wizard of Oz User Interaction Studies

6.1 Abstract

This paper describes WOzOS (Wizard of Oz Operating System), a Java-based platform that can be used in multimodal Wizard of Oz (WOz) experiments. In addition to the platform design, a study using WOzOS is described. Human-machine interactions in a multimodal environment were the focus of this study. WOzOS was used to collect multilingual (English, Portuguese and Swedish) data for research purposes, including for training of a multimodal fusion/fission service.

6.2 Introduction

In the Wizard of Oz (WOz) paradigm, a user interacts with what appears to be a fully functioning automatic system. Unbeknownst to the user, however, the system responses are generated in real time by human operators who remain unseen (Figure 6.1). In this manner, a variety of user interfaces can be prototyped, and user responses to a range of system behaviours can be studied. Such simulation-based research is useful in the design of multimodal interfaces Oviatt et al. (1992) as well as for the collection of multimodal corpora Yang et al. (2000). Multimodal data gathered in this manner can guide system design and help in optimizing human-computer interaction.

In many cases, WOz systems are purpose-built to investigate specific systems (Oviatt et al. (1992), McInnes et al. (1997), Wyard & Churcher (1998)). Two research groups have tried to provide more generic platforms, which would allow simulation-based research in a wider variety of contexts. In NEIMO Coutaz et al.

(1996), Apple's *HyperCard* was used to assemble an UI in real time. The system allowed both GUI elements and video, but not sound, to be transmitted and logged. In SUEDE Klemmer et al. (2000), a speech interface could be generated from a relatively simple toolkit. WOzOS represents a further contribution to the set of tools that can support simulation-based research across a variety of situations. It combines standard GUI elements with spoken input and synthesized voice output. In this paper, we describe the basic WOzOS architecture, and report on a study we conducted, in which multilingual, multimodal data was collected for research and development purposes.

Figure 6.1: Wizards (behind the scenes) and Client (oblivious to deception)

6.3 The WOzOS Platform

6.3.1 Hardware

WOzOS runs on three networked workstations and comprises three separate applications: Operator, Session Manager and Client. Two of the workstations are "Wizard" stations (Session Manager & Operator), the other is the Client interface. Deception is the principle underlying the experiments, so the Wizard stations should be hidden and, preferably, should not be in the same room as the subject. The platform includes a commercial text-to-speech (TTS) module (ScanSoft's *RealSpeak*) for generation of synthetic system spoken output, but this could readily be replaced by an open source TTS module (e.g. *Festival*). In addition to a workstation, the Client also uses a head-mounted microphone and headphones.In the following description, the application used in our corpus collection will be used as an illustrative example. In it, the Client interacts with a Personal Information

Manager (PIM), which provides a subset of Microsoft's *Outlook* functionality, augmented by speech output.

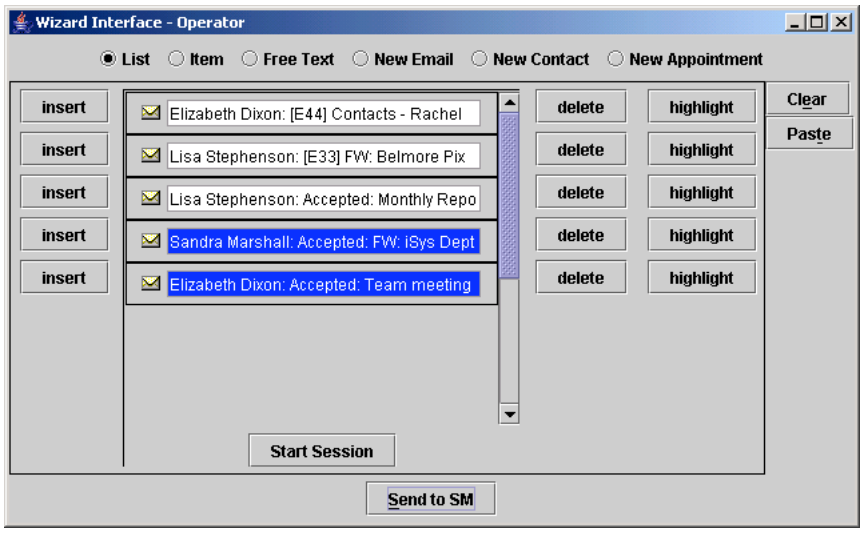


Figure 6.2: WOzOS Operator application

6.3.2 Wizards

The example domain requires the rapid construction of appropriate system responses (graphical and TTS) in real time based on Client requests to the system. A credible system response must be fast; excessive delays could influence subjects' impression of whether they are interacting with a computer system Oviatt et al. (1992). In WOzOS the general target was a response time of a few seconds at most, this target was achieved; average response time[1] of a typical, sample session was 5.5 seconds, with standard deviation 4.7. To facilitate a quick response the Wizards compose their GUI responses using templates and widgets that can be included or excluded. WOzOS can simulate adaptive systems, where human performance is assumed in all decisions related to how to adapt. In our case, this means that Wizards choose from a constrained set of user interface elements (widgets on the screen, words and phrases by voice), in order to compose the output. The choice Wizards have in reacting makes them a study subject also, and allows for studies of adaptivity, provided the situations triggering adaptation (noise, physical activity) are simulated.

[1]Response time being defined as time difference between end of Client input and start of system (Wizard generated) response.

The Wizards assume two roles: the **Operator** retrieves data from an application such as *Outlook*, and prepares the visual representation. In parallel, the **Session Manager** assembles the voice output and ensures that all experimental tasks are carried out. On-line chunking of the data in task sets has proven helpful in navigating the data created.

Operator

The Operator's interface is used to assemble screen content appropriate to the experimental context and Client requests. The Operator can copy content from other application sources and paste it into a screen area. This can then be further edited before sending to the Session Manager.In the PIM context, within which we have experimented using WOzOS, content like e-mails or contact data may be copied from *Outlook*. The Operator's application automatically arranges the content either as a list (Figure 6.2) or as a single item. The data from this initial study was used for design of a multimodal PIM application (FASiL project), which is to run on a mobile device e.g. PDA. For this reason the output screen area was kept to a size consistent with such a platform. There is no reason why screen size has to be limited. Indeed, following further development we hope to demonstrate WOzOS in different contexts, using differing sources for content and output screens. To aid and speed up the Wizards' work all application functions have hotkeys associated with them.

Session Manager

The Session Manager application controls the sequence of the experimental session and the output presented to the Client. The Session Manager can set the experimental environment (output modalities presented, TTS/noise volume, background noise), compose TTS utterances and add additional screen elements (buttons) as appropriate to output.

The Session Manager receives screen content (Figure 6.3) from the Operator and has the option of editing and/or augmenting this as desired. In parallel, the Session Manager can compose TTS utterances to send to the client. TTS utterances can be sent with screen content or independently. TTS utterances are often used as

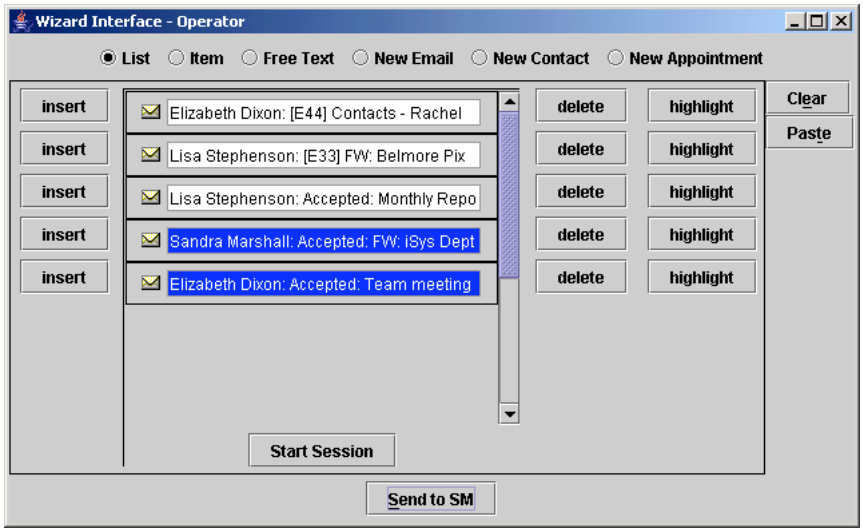


Figure 6.3: WOzOS Session Manager application

system ‘pacifiers’ to mask delays in Wizard responses. This TTS-only output can also be used where a unimodal (voice only) context is appropriate. We can add background noise if required by the experiment.

6.3.3 Client

The Client Application receives the graphical user interface dialogs from the Session Manager (Figure 6.4). Running on a Tablet PC, the subject can use the GUI on the touchscreen. Text input aids (handwriting recognition, on-screen keyboard) could be used, though they were not appropriate for our experiment.

All Client interactions with this application screen (mouse clicks/drags, scrolling, text input) are displayed immediately on the Operator (Figure 6.5) and Session Manager application screens to allow the Wizards to monitor the Client interaction.In addition, screenshots showing Client interaction are generated and sent to Session Manager for storage.

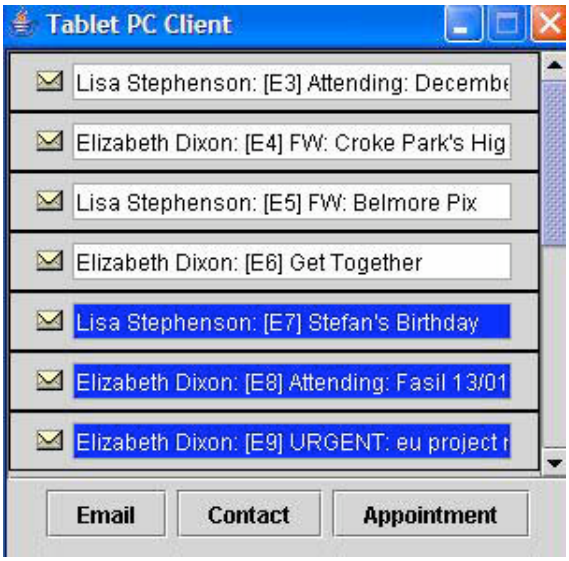


Figure 6.4: WOzOS Client application

6.3.4 Text-to-Speech

The TTS utterances are output via the Session Manager workstation. To speed up response times and to maintain consistent system utterances, we found it useful to define a set of utterances and to encode these in an abbreviated form, e.g. “pw” is expanded and uttered as “Please wait, I’m checking this”.

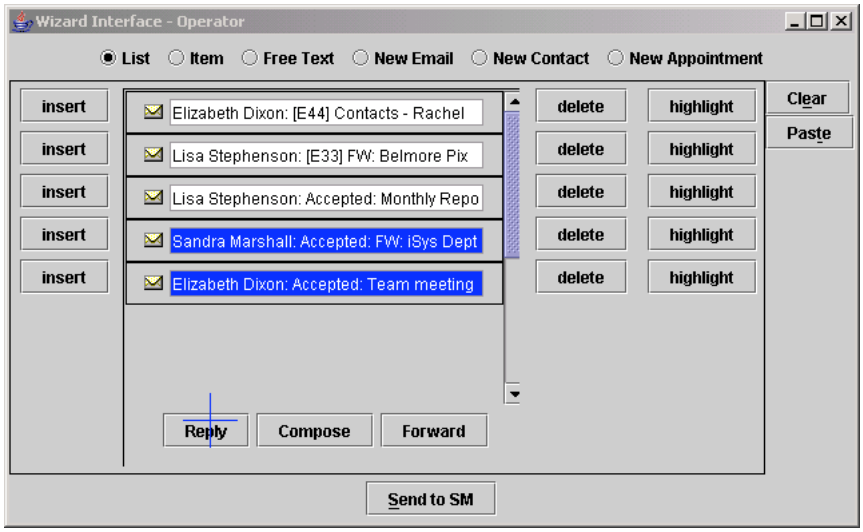


Figure 6.5: Client interaction (click on “Reply”) is displayed

6.3.5 WOzOS session data

All session data are logged on the Session Manager workstation. Time-stamped events of Client and Wizard actions are logged, along with an audio recording of both TTS utterances and Client speech. Screenshots of the screens sent to the Client, and their interaction with the screens are saved and referenced in the log. We provide a tool chain to generate XML data in the *TASX*[2] format from the raw log files generated by WOzOS.

6.4 Multimodal Study using WOzOS

WOzOS was developed as part of the FASiL[3] project. The R&D vehicle of this project is a multimodal, multilingual PIM application, suitable for use on a portable platform, which is developed by a consortium of European partners.

6.4.1 Study aims

In general a particular focus of our research has been the coordination of the graphical and voice outputs, which are tailored to meet the demands of a specific situation and user Reitter et al. (2004).In line with this and the particular objectives of the FASiL project, our study had two separate goals. Firstly, we wished to gather a reasonably large corpus of user interaction with a Virtual Personal Assistant (VPA), which gave access to email, calendar and contact information. This corpus was to be used in training machine learning modules responsible for dialogue management (grammar induction, etc.). Our second goal was to supply an empirical basis for the evaluation of algorithms for multimodal fusion, as well as fission, i.e. the generation of multimodal system output, where models of coherence and pronominalization in multimodal human-computer dialogue play a role. This second goal required the design of a formal experiment in which both available modes and user situation were controlled variables. The SmartKom Turk (2001) project had similar motivations[4] for conducting a large multimodal WOz

[2] http://tasxforce.lili.uni-bielefeld.de

[3] Flexible and Adaptive Spoken Language and Multi-Modal Interfaces

[4] SmartKom also uses gestural input, so video of interaction was annonated as part of the corpus. WOzOS presently restricts itself to screen interactions.

study, however this was only done in one language: German. Our study was carried out in three languages: English, Swedish and Portuguese. As a first step, we had to devise a suitably constrained set of tasks which novice subjects could be required to complete. To this end, we ran a unimodal (voice only) pilot study (70 subjects, three languages), which helped to refine our subsequent task design.

6.4.2 Study description

Three sites (Ireland, Sweden, Portugal) each conducted the study using 30 subjects recruited locally. Subjects were professionals and students (average age: 34.6, standard deviation: 10.6), had prior experience with personal information management software such as *Outlook*, but no specific training in spoken user interfaces. Subjects completed a fixed list of tasks such as "arrange a meeting with Veronica for tomorrow at three to discuss the new recruitment procedures". Each subject completed tasks in a unimodal (voice only) and multimodal condition and in the presence/absence of a noise distractor. Wizards received training; training materials as well as subject tasks were standardized across experimental sites. The detail of the experimental design and results will not be further discussed here.

Subjects found the system easy to use, and were not, in general, aware that they were interacting with people, rather than an automaton. This is somewhat surprising as the system offered near-perfect speech recognition and understanding, which is beyond the abilities of any state-of-the art system.

6.5 Conclusion

We have described a new Wizard of Oz system that can facilitate the collection of user responses to a simulated system. WOzOS supports both screen-based and voice input and output. Two Wizards collaborate to assemble content and system responses on the fly, allowing for a rich set of behavioural data to be collected. We have used WOzOS to collect interaction data in three languages, and will shortly release the resulting annotated, multilingual corpus through the FASiL consortium (www.fasil.co.uk). One of our FASiL research partners, The Royal National Institute for Deaf People (UK) also ran a parallel study on a large number of hearing

impaired subjects of mixed backgrounds and ages. The study design was similar to that outlined above, but obviously had to be adapted for subjects with significant hearing loss. This corpora and WOzOS itself will also be made publicly available.

6.6 Acknowledgements

We would like to acknowledge the hard work done in putting together the platform and conducting the study. In particular Nathalie Richardet, Erin Panttaja, Stefanie Richter & Wei Zhu (Media Lab Europe), Roman Zielinski, Sara Holm & Per Idoff (Cap Gemini, Sweden), Nuno Beires, Luis Almeida; & Rui Gomes (PT Inovação, Portugal), Guido Gybels & Jamie Buchanan (RNID, UK) and the team at ScanSoft.

Chapter 7

Evaluation

Evaluation, as we have seen, is a very difficult proposition in user interface design and implementation. Originally we had planned to integrate MUG Fission and Fusion into VPA 2 and test them as part of the full system evaluation. Implementation constraints have made that impossible. In the absence of a full end-to-end system, we have designed an alternate evaluation methodology, as described in this chapter. The evaluation itself will be completed later this summer.

The Evaluation of Adaptable Multimodal System Outputs

Abstract

Adaptable multimodal systems are difficult to test. We present a methodology for evaluating parallel multimodal output which is generated in response to a specific set of user, device, and situation constraints. We focus on the generation of multiple variants of user interfaces for small-screen graphical devices with natural language voice output, within a system we term UI on the Fly. Our methodology tests any system that ranks potential output variants using a fitness function.

7.1 Introduction

Coordinated multimodality, adaptivity, and automatically-generated interfaces are relatively new paradigms in human computer interface design. Rather than sequentially employing modes to convey information to the user, several modes are used redundantly or complementarily. Sound-enabled interfaces are a simple example for coordinated multimodality, voice-enabled SALT[1] documents another. Some research prototypes represent dynamically generated user interfaces, which can be adapted to the user's special needs in a given situation, for example, if the user cannot pay much attention to the screen while performing maintenance operations or driving a car. Natural language plays a central role in such interfaces, not only in voice output, but also in visual user interfaces adapted to devices like mobile phones that have only limited input options.

[1] Speech Application Language Tags, *www.saltforum.org*

These advanced systems are notoriously difficult to test, as they change their behavior dynamically and unpredictably. As systems begin to follow new interface paradigms, evaluation metrics will need to take into consideration additional learning time on the part of the user. In addition, test systems are often limited in their functionality, and may depend on the implementation of a complete dialogue system that may not be available during testing. We circumvent some of these problems by focusing on a system with an *adaptable situation model* which remains fixed during each test case.

There are many different measurements for describing a 'good' system. Does it function within the specification of its design document? Can it be used by its target group? Is it accessible to the hard of hearing? To the blind? To those with motor impairments? Even accessibility is hard to define. A system may be technically accessible without being usable. Does it allow users to complete the tasks they set out to complete? Are these tasks useful in their daily lives? Do they enjoy the system? Do they trust the system?

In our example case, the system performs the automated, parameterizable generation of a user interface with a visual component and text-to-speech voice output for sending email (see Figure 7.1). The system relies on a grammar of hierarchical components to define the display. The generation algorithm and the components ensure that the output is consistent across multiple devices. The design choices that the algorithm makes are also based on the prediction of utility and cognitive load that a possible output variant will have. We describe the underlying formalism in Section 7.4.

In what follows, we will present a general methodology for the evaluation of multimodal system outputs which we believe is capable of potential application to a wide variety of evaluation problems. We illustrate the method as it is currently being applied to a specific application (an email client), along with a concrete formalism which easily supports the method by the generation of multiple output variants. Full evaluation results will be presented at the workshop.

(a) (b) (c)

Figure 7.1: a) Voice: “Send the email regarding Irish Weather?” b) Voice: “Send the email?” c) Voice: “Do you want to send the email to Mick regarding Irish Weather?”

7.2 Recent work

It can take a very long time, on the order of years, to find out if users will really use a system or accept a new paradigm. This acceptance may be dependent on (or impeded by) other factors (e.g. issues with documentation, trust, advertising, cost...) Reiter & Dale (2000). It is not surprising that a user who has years of experience using a two-dimensional graphical user interface with a keyboard and a mouse will seldom find that a novel interface with 3D graphics and coordinated natural language interaction is a better way to input commands and data, at least at first.

In many projects related to natural language or multimodal dialogue, evaluation is ignored altogether. Experiments with human users are often used mainly as part of the system design process. (as in Feiner and McKeown, 1988). Many of these systems are research prototypes that apply to a limited domain or a limited number of interesting test cases. A user-based evaluation is only feasible once the system is sufficiently stable to allow users to access it over time. While this is an eventual goal, preliminary evaluation will prevent wasting time on substandard user interfaces.

When it comes to the evaluation itself, there are a variety of quantitative measures (time to perform, accuracy, percent agreement of assessments) and qualitative ones (user perceptions of utility, ease of use, and naturalness). Maybury & Wahlster (1998)

Qualitative measures also include the study of think-aloud protocols and observation of users. These techniques obviously require a stable and even robust system to be available. At earlier design stages, a cognitive walk-through or a heuristic evaluation against rules-of-thumb Cockton et al. (2002) can provide guidance.

Evaluation based on user models employs a simulated user that behaves under, ideally, the same limitations and strategies that a human user would demonstrate. GOMS (Goals, Operators, Methods and Selection rules, Kieras, 2002) is a methodology that allows the formalization, even before a system can be used, of elements of a user interface in terms of the knowledge required from a user. A GOMS model seems inappropriate for an adaptable system that may dynamically change the operators available to the user. *Adaptivity involves a constantly changing system model*. Its benefits become clear only in the context of a user under certain external limitations - such as those imposed by parallel, unrelated tasks like driving a car or participating in a conversation.

In SUPPLE, Gajos and Weld (2004) present a system that adaptably generates a graphical user interface. They discuss a number of different evaluations to their system. Efficiency tests of the generation algorithm show the effect of certain proposed optimizations. Gajos and Weld propose judging the quality of the user interfaces by comparing the system's decisions to those made by human designers under similar constraints regarding the available user interface widgets.

In general, subjective testing asks a user or designer for their impression and judgment of a system. Reiter and Dale (2000) discuss having experts evaluate both automatically-generated and hand-generated examples. In Comfort, Knight et al. (2002) evaluate wearable UIs on the bases of emotion, attachment, harm, perceived change, movement, and anxiety. This set of criteria was generated by multidimensional scaling, and could be adapted for use with other mobile (but not necessarily wearable) devices.

The NASA-TLX system Hart & Staveland (1988) is a measure of subjective workload. It has users rate a human-machine environment based on mental demands,

physical demands, temporal demands, their own performance, effort, and frustration. A weighted superposition of these features, based on relative ratings given by the user, leads to less between-rater variability than do one-dimensional ratings.

Direct testing compares metrics that are directly related to the interface itself, such as task completion time or success rates. Walker et al. (1997) score dialogue systems with a combination of dialogue success measure and various utterance-related costs. Dialogue success depends on whether slots for a dialogue are correctly filled. Costs for normal utterances and repair moves are counted separately and, like the dialogue success, are weighted using multiple linear regression, with user satisfaction as an external factor. The advantage of this approach is that dialogues may then be scored without an explicit user judgment.

Beringer et al. (2002) modify the framework significantly in order to evaluate free dialogues with their multimodal system, where users are given a much less specific task which cannot be described in terms of necessary and optional slot-filler pairs.

Indirect testing examines things like walking speed or ability to concentrate on outside tasks. Pirhonen et al. (2002) use the percentage preferred walking speed that a user is able to maintain while using the device to evaluate usability.

When doing evaluations, it can be very difficult to compare results from different systems Bontcheva (2003). It is important to ensure that both the baseline and adaptive versions of the system are generating in real time.

7.3 Evaluation

In this section, we propose an evaluation methodology. We expect very similar methods to be applicable to a range of dynamic human computer interfaces that generate output based on a number of constraints and define a fitness function to rank solutions. This is true whether or not these systems are multimodal, and without regard to the degree or specific instantiation of the multimodality.

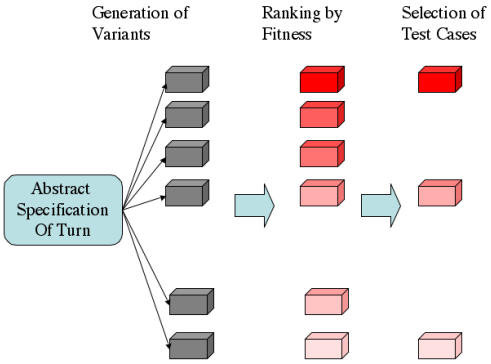


Figure 7.2: Generating test cases for user evaluation

7.3.1 Method

Our method expects the definition of a candidate fitness function with which multiple candidate output variants can be ranked. The fitness function estimates the projected utility of a variant depending on factors defined by the system designer, and is intended to capture the relevant features of the dialog context, device constraints, user preferences and situation-specific elements. Figure 7.2 illustrates the process whereby an abstract specification of the dialog turn is received from the dialog manager. This is used to generate many candidate output variants which differ in their informational density and the distribution of information across modes. The fitness function ranks these from best to worst, allowing well-differentiated test cases to be selected from among the best, middle, and worst cases for user evaluation in an experimental situation.

Without a gold standard generation system for dynamic multimodal user interfaces to compare against, controlled user trials will allow us to evaluate the usability of the interfaces we create. Key to our approach is the use of sufficiently discriminable output variants (as provided by the ranking) to ensure that we capture a range of user reactions, and can thus gauge the suitability of the fitness function.

Multimodal output generation has been the focus of various grammar-driven generation algorithms, such as COMET Feiner & McKeown (1998) and SUPPLE Gajos & Weld (2004) which optimize text and graphics layout in documents for print or display on various screens. In dialogue systems, coordinated multimodal-

ity can be found in some embodied conversational agents (e.g. Cassell et al., 2000; Wahlster, 2002). Essentially, these systems form intelligent multimedia-interfaces.

7.3.2 A note on adaptable systems

We make a distinction in this paper between adaptive systems and adaptable ones. Both adaptive and adaptable systems present novel challenges, as user expectations may be confounded, and interface consistency needs to be maintained despite variation in surface realization. We see a role for adaptable systems where an *information bottleneck* arises, e.g. because of the use of a small screen device, situational constraints, or changing user preferences. In these cases, we address the problem of adapting the output to the situational demands by generating multiple variants, and selecting among them based on a fitness function which takes these constraints into account. Adaptive systems, on the other hand, change over a longer time scale to match the user's (or group of users') needs or skills. Our current methodology does not involve sufficient testing time to experiment with such adaptivity, though it could be modified to do so.

7.3.3 System of ranked variants

Prior to the design of the system, we identified several areas where we can parameterize the output. The *device model* specifies capabilities of the end-user devices, in particular the screen size and interaction options such as a touch screen or variable buttons, as used in many cell phones. The *user model* reflects preferred multimodal interaction (and signal integration) patterns.

The *situation model* reflects external constraints imposed on the interaction with the device. These constraints originate from ambient noise, the users' cognitive workload, manual workload (as in cooking, driving), and sensory workload (watching a movie, walking, listening to a talk).

Our evaluation method controls the adaptation models in order to reflect carefully chosen real-life situations. The more adaptation parameters there are, and the more values that are under consideration, the greater the number of experiments needed to gain sufficient data to show a significant effect of the system's design

choices. Over long periods of time, user model adaptation can be problematic, as the system and user may adapt to each other reciprocally. For these practical reasons, we decided to vary only the situation and device models.

In an adaptable system in which multiple variants are generated and scored, the scoring metric (see Section 7.4.2) can be tested by creating versions of the system for the user to interact with. Each version is based on a particular user model, situation model, and device model, and compares the best-rated, worst-rated, and, optionally, one mid-ranked option (according to the fitness function) for each situation. In this manner, the fitness function can be evaluated, as a high degree of discriminability among the variants presented to the user is assured.

Both subjective and objective measures of interface usability can then be used to assess whether the fitness function can boost user satisfaction with a given output variant and, indeed, whether adaptivity is of advantage at all to users in a specific situation. Task completion times, task completion rates (recognition of incorrect system responses), user frustration levels, and user satisfaction are all candidate variables for evaluating the fitness function.

7.3.4 Scales

There are several different scales on which one can measure a given test. A scale may be absolute or comparative. It may test things subjectively, directly, or indirectly. It may compare different instantiations or different underlying reasons for adaptivity.

In absolute testing, we ask “is this a good user interface?” This is a difficult question to answer. In general, the testing of a user interface comes down to comparing it with other systems. Sometimes that means comparison with similar interactions between humans, and other times it may mean comparison with the behavior of a simulated system in a Wizard of Oz scenario (See Section 7.3.5).

In comparative testing, different output variants or different versions of the same system are compared to find the relative merits of the systems in the eyes of either users or human designers. This tends to be easier to control, as it can be ensured that the systems are, in fact, comparable.

Even multimodality has multiple scales. A user may use the screen or sound for output, and may use touch screen, keyboard, or voice for input. Multimodal interfaces may also use gesture, haptics, or even smell as a mode for interaction.

Stressors on the user may be internal, as when the user is trying to pay attention to a meeting while checking for an emergency email message, or external, as in a noisy restaurant where one cannot escape distraction. A user may have limits placed on them, based on how public or private their setting is, which may be changed significantly by personal and cultural issues.

User distraction levels and different device models are not, in themselves, applicable to every multimodal system, but each system will have its own set of constraints that will be used to define the output variants generated and the fitness function used to select the optimal variant.

An ideal system evaluation would test each relevant metric individually, but there is a need for testing that will give an overall judgment of an interface.

7.3.5 WOz

Wizard of Oz (WOz) testing has an important role to play in the creation of adaptable systems. It can give an indication of what kinds of interfaces are needed and how those interfaces will be used without the initial cost of building a whole system. However, over-reliance on WOz testing can be dangerous: some aspects of a WOz simulation may not be replicable in the actual application (e.g. near-perfect speech recognition). In the special case of evaluating adaptable systems, it can be difficult to ensure sufficient consistency in the work of the wizards to ensure that the only differences between trials are those demanded by the adaptation.

7.3.6 Methodology

We create situations in which we can limit the user's attention to various modalities and collect information on user satisfaction using the NASA-TLX scale Hart & Staveland (1988), task completion time, and task success rates.

For mobile systems, the ability of the user to use the system even when distracted

is key. To this end, the testing will involve the user being distracted from the requested task. Undistracted usage would parallel a user at his desk or working in some other quiet, non-distracting environment. This situation could serve as a control. For a system which includes a screen display with auditory output and pen and voice input, one form of distraction would be auditory in nature, as that found in a crowded restaurant, while listening to the radio, or while in a meeting. Visual and tactile distractions would be those found in a meeting, while cooking, or while walking down the street. These situations, of course, must be customized to the aims of the particular modalities present in the system in question.

We divide the testing into two phases, for ease of understanding. The first phase tests the fitness function's ability to choose the best of the interfaces for a given situation. This would mean selecting (see Figure 7.2) the best, middle, and worst cases for each situation.

The second evaluation phase allows users to use the ideal variant for each situation in other situations. This means evaluating whether the fitness function really does select the optimal design for each situation correctly, as well as determining whether there are distinct ideal adaptations for each situation.

These two sets of tests are very similar. In most cases, the total variant list for each scenario will be the same. But the worst-case interface for a user who is subject to auditory distraction may be an unequivocally bad interface, rejected by both users and the fitness function.

In the next section, we describe the application of this methodology to a specific case study: UI on the Fly.

7.4 UI on the Fly

In this section, we outline our multimodal generation system, which has a grammar and a fitness function at its core. The system is currently undergoing a full evaluation.

7.4.1 MUG

Multimodal functional Unification Grammar is a non-deterministic grammar formalism Reitter et al. (2004) that generalizes decisions about how to deliver content in a multimodal user interface. A grammar in this formalism specifies an adaptable user interface. The formalism is an extension of Functional Unification Grammar (Kay, 1979; Elhadad & Robin, 1992) that ensures content coordination in the different modes. It allows for the generation of multimodal user interfaces.

The application of a MUG yields several solutions that are faithful to the original specification and consistent and coherent across the different output modes. Only one of these solution is considered the best one – according to a fitness function, which incorporates the user, situation and device models.

We demonstrate MUG in the context of a limited-domain user interface for a mobile personal organizer (see Figure 7.1).

A MUG is a set of *components*. Each of them specifies a realization variant for a given partial semantic or syntactic representation, similar to a rule in a production grammar. The components are attribute-value structures. The generation algorithm chooses components from the grammar and unifies them iteratively with the original input specification, thereby instantiating several layers of output planning and surface form realization.

While allowing for cross-modal consistency, the attribute-value matrices allow us to distinguish information a) that needs to be shared across all output modes, b) that is specific to a particular output mode, or c) that requires collaboration between two modes (for example, deictic pronouns).

There may be several competing components in the grammar for a particular job, all of which unify with a given partial semantic input. This translates to a design choice the system has to make. Design choices are never made individually. They often depend on other choices. For example, choosing to render the full subject line of an email in a display variant on a small screen device might not leave enough room for the (more important) name of the recipient. The system therefore evaluates the variant as a whole[2].

[2]Practically, best-first / A* search algorithms may be used to optimize the search for an optimal solution. But that has no consequences for the evaluation.

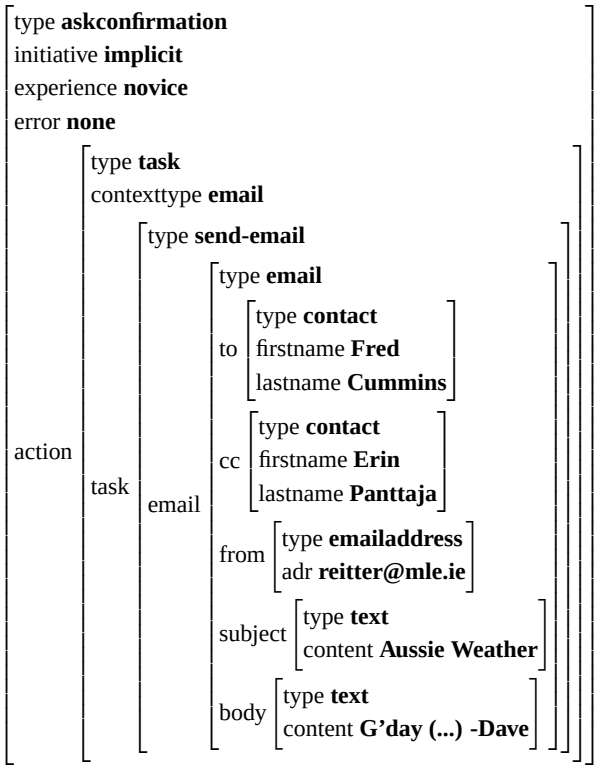


Figure 7.3: Input representation: confirmation of sending of an email. E-Mail body text abbreviated.

MUG enables some feedback to the dialogue system about which parts of the dialogue semantics were actually realized in a given situation, as addressed by Wahlster (2002). A mode-specific attribute (*realized*) is instantiated by the grammar for each semantic entity that has been incorporated in to the output in the given mode.

Design variants are ordered according to the outcome of a fitness function. The best variant is is that optimally adapted to the given situation, user, and device (see Section 7.4.2).

7.4.2 Fitness function

Finding the best solution to the hard constraints defined by the grammar can be seen as optimization problem. What do we optimize?

There are different approaches to formulating the scoring function, and usually there are several considerations that are weighted. In SUPPLE (2004), Gajos and

Weld predict the effort a user has to make in order to reach each element of an interface. Such a user-model driven fitness function still leaves the designers with many choices – for example, whether the cost for each user interface element should also depend on the maximum-likelihood probability for its actual use.

In UI on the Fly, we generate simpler, but multimodal interfaces for small-screen devices. The number of elements shown on a screen is small, and the user interface widgets defined by the MUG do not differ greatly in the time it takes to operate them. We see major cost differences, however, in the degree to which the voice modality is used (it takes time to listen to system speech). We therefore model the utility of a particular multimodal output as a competition between reading / listening time and the benefit of presenting important information.

By default, we try to be as helpful as possible, with information that is deeply embedded in the semantic structure receiving lower priority than higher elements. Redundant information, that is, information that is presented in both modes, does not receive a double benefit. Information that needs to be presented according to the assumed dialogue management component leads to a heavy penalty if it is left out during generation stage.

The trade-off lies in the cost of the output, which is estimated in terms of the cognitive load imposed on the user, who needs to read new text on the screen or listen to the voice output.

These constraints are formalized in a score that is assigned to each variant ω, given a set of available Modes M, a situation model $< \alpha, \beta >$, and a device model ϕ:

$$s(\omega) = \lambda \sum_{<e,d> \in E(\omega)} u(e,d) - max_{m \in M}(\beta_m t_m(\omega))$$

$$u(e,d) = P(d, \sum_{m \in M} (\phi_m \alpha_m e_{m|realized}), e_{realize})$$

The first part of the sum in s describes the utility benefit. The function E returns a set of semantic entities in e (substructures) and their embedding depths in d. The function P penalizes the non-realization of requested (attribute *realize*) semantic entities, while rewarding the (possibly redundant) realization of an entity. The reward decreases with the embedding depth d of the semantic entity. (Deeper entities

give less relevant details by default.) The request is encoded in the *realize* attribute and the actual realization feedback is given in the mode-specific attribute *realized*. α_m are benefit and β_m are cost coefficients that represent the user's attention level in each mode.

The cognitive load (second part of the sum) is represented by a prediction of the time $t_m(\omega)$ it would take to interpret the output. This equals the utterance output time for a text spoken by the text-to-speech system or an estimated reading time for text on the screen.

The utility/time normalization coefficient λ can be manually estimated or learned from a corpus. If the evaluation setup is used, λ will be acquired from a separate training partition of the data.

7.5 Evaluating UI on the Fly

In carrying out preliminary evaluations of UI on the Fly, we need to bear in mind that a) it is not a complete dialogue system, but b) it should be evaluated with human subjects.

One of the ideas behind UI on the Fly is that local decisions about the generation of multimodal output may incur a local cost, but benefit the dialogue. For example, a certain output may contain more information and thus be longer. But in turn, the system may be able to remove an additional confirmation step. Such decisions can only be evaluated in the context of a full dialogue.

To evaluate whether the prediction of cognitive complexity is realistic, we will measure task completion time for a predefined task that involves sending an e-mail. We will compare the performance of a system that chooses the output variant deemed optimal against one that always chooses a mid-ranking output variant.

7.5.1 Recreating usage situations in the laboratory

In an attempt to broaden the range of respondents in this evaluation, it will be built as a web page to be used in the user's own office or home. This will allow us to

test the system with a variety of different computer-literate users.

The evaluation of the system in the laboratory attempts to recreate the important mode-specific characteristics of a range of hypothetical situations.

The users will be given a computer-game task as an auditory, visual, and tactile distraction. This will be a flash program in the testing web page. In order to ensure that the user is paying attention to the game, their score will be recorded. The time they are allowed for each turn will also be limited.

This will not exactly mirror the target task of walking down a busy street, but will simulate some of the distraction and cognitive load.

7.5.2 Devising tasks

Each user will be asked to send one email message using the system, while performing the distraction task. This message will be selected from a bank of three messages. They will not be using a full dialogue system: each turn of the task will be represented by an output turn from the system, then a corresponding input from the user. Errors by the user will be ignored by the testing system (though recorded for the evaluation).

The tasks will all be web site-based, but will simulate usage of either a small screen cell phone or a PDA-device (see Figure 7.1).

Each task will involve approximately five system turns, and the two selected variants will be the first and middle option from a pool of 30-90 variants created by the system for each turn.

Some of the test turns will involve mistakes on the part of the system. Whether or not the user catches these mistakes will be recorded as the user's error detection rate.

There will be three different tasks, and two different devices. There will be 10 users of each task, for a total of sixty users.

7.5.3 Measuring quality

We will be collecting several different kinds of information from each user. We will start with a user questionnaire, to establish their background and whether their system is sufficient for the experiment.

For each system turn, we will record the task completion time and whether the task was completed successfully.

After the task, we will ask the user how appropriate the system output was in the given situation (user satisfaction). By pairing user satisfaction ratings for different utterance types we can show whether the fitness function and the user satisfaction data show a significant correlation, and whether the situation-specific adaptation has a significant effect on the user satisfaction.

7.6 Conclusion

We have discussed several approaches to the evaluation of adaptable multimodal dialogue systems and their output generation components. We have presented a case study giving a preliminary view of the evaluation of a concrete instantiation of such a system under realistic constraints. Meaningful evaluation, even of a single subsystem with limited functionality, is feasible.

This methodology can be applied to any system that uses a fitness ranking to choose an optimal interface to present to a user. Each parameter (situational, user, or device) added to the system will, of course, increase the number of tests (and users) required, but each additional constraint can be easily be compared against tests already completed.

Acknowledgements

This research was in part funded by the European Commission under the FASiL project, contract number: IST-2001-38685.

Chapter 8

Dissemination Activities

8.1 Overview

The research activities described in this report were publicized on many occasions – in publications and public talks (listed below) and at Media Lab Europe's *Open Houses*, which are events held in Dublin for an international audience of decision-makers, journalists and scientists. Further talks (not listed) were given at Media Lab Europe events for invited audiences. Catering for a broader public audience, three MLE-based, partially FASiL-funded researchers were featured in MLE's 2004 *Innovators* brochure.

8.2 Journal articles

David Reitter. Simple signals for complex rhetorics: On rhetorical analysis with rich-feature support vector models. LDV-Forum, GLDV-Journal for Computational Linguistics and Language Technology, 18(1/2):38-52, 2003.

8.3 Conference and Workshop papers

David Reitter. A development environment for multimodal functional unification generation grammars. In *Proc. Third International Conference on Natural Lan-*

guage Generation (INLG04), 2004.

David Reitter, Erin Panttaja, and Fred Cummins. UI on the fly: Generating a multimodal user interface. In *Proceedings of Human Language Technology conference 2004 / North American chapter of the Association for Computational Linguistics* (HLT/NAACL-04), 2004.

David Reitter and Manfred Stede. Step by step: underspecified markup in incremental rhetorical analysis. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora* (LINC-03) (at EACL 2003), Budapest, 2003.

8.4 Thesis

David Reitter. *Rhetorical analysis with rich-feature support vector models*. Master's thesis, University of Potsdam, 2003.

8.5 Talks

David Reitter. *UI on the Fly: Generating a multimodal user interface with functional unification grammar.* SALS-SIG Research Seminar, Macquarie University Sydney, Australia, December 2003

David Reitter. *Aspects of Generating a Multimodal User Interface.* Speech Seminar, Interactive Systems Lab / Language Technology Institute, Carnegie Mellon University, Pittsburgh PA, U.S.A., July 2004

David Reitter. *UI on the Fly – a changing multimodal user interface interface for mobile people.* Media Laboratory, Massachusetts Institute of Technology, Cambridge MA, U.S.A., July 2004

8.6 Multimedia

David Reitter, Michael Cody: *Multimodal user interfaces in the FASiL Virtual Personal Assistant.* Video clip, 0:53. available at: http://www.medialabeurope.org/asi/fasil/

Appendix A

Multimodal Integration and FASiL VPA 1.5

FASiL VPA 1.5 was defined at the multimodal workshop in London as a proof of concept for the FASiL multimodal architecture and an opportunity to integrate the work that VOX and MLE have been doing in a less ambitious manner than the full VPA 2 functionality. It is intended to show end-to-end functionality in the context of the send email task only. Our original goal is a VUI similar to that of VPA1, with a simple GUI with few functional elements. We are not trying to recreate Outlook. The original plan for translating VPA 1.5 to Portuguese has been dropped for lack of resources.

The original schedule set at the time was to complete development by the end of January. We agreed at the time that this was optimistic, and subsequent developments mean that this deadline will be missed.

VOX and MLE met in November to define the interfaces between the various components.

Vox implemented much of the MM Gateway, and a first version of the fission module has been integrated. MLE created have a simple MUG that generated the existing prompts and GUI text for some parts of the email system, with a goal of covering all of email functionality.

Fusion required additional work to implement the interface with the MMGateway, and work with a modified semantic interpretation. There was a problem with providing the required timing information to Fusion (start-of-speech event and timings in the semantic interpretation).

A working VPA1.5 prototype was scheduled for delivery in February. Ideally this would have covered the whole of send-email. We identify a contingency whereby we cover a subset of it.

As of the consortium meeting in January, this integration process was cancelled in favor of separate work by Vox on VPA 2 and by MLE on independent Fission and Fusion services.

Appendix B

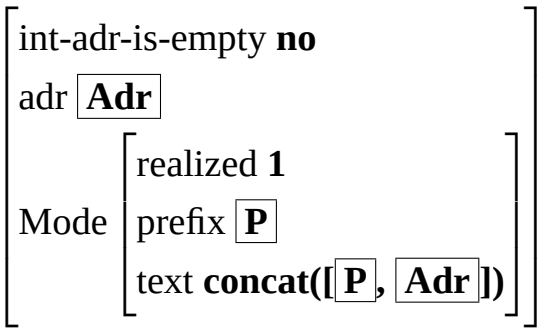
Grammar Examples

B.1 Component variants

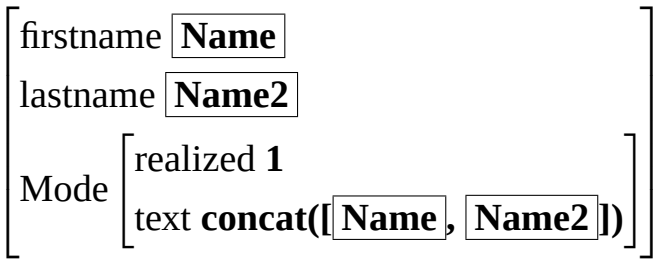
There may be several different components defined for displaying a particular type of data. Some may be appropriate for specific types of situation (or require certain information to be present). Others may be instantiated in any situation, but will change the score of interface which is presented to the user in the end.

There are a wide variety of different ways to display a contact's name.

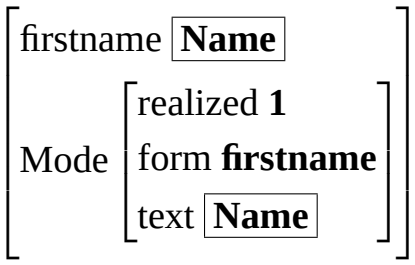
If we know their email address, we can display that.



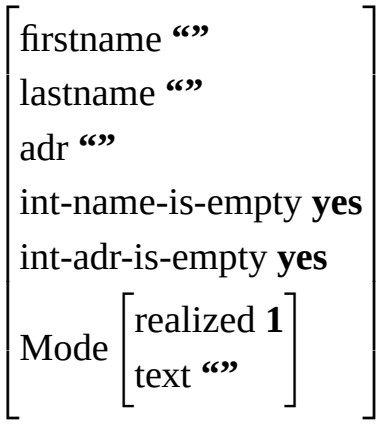
If we know their given and surnames, we can concatenate those together.



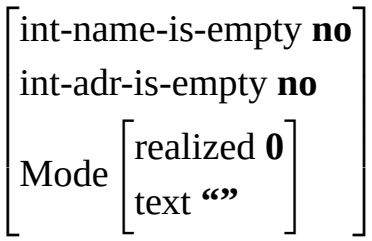
In some cases, we want to give only their first name, for brevity.



If we don’t have a name or an email address, we won’t display them, but (note that realized is 1) we incur no penalty.

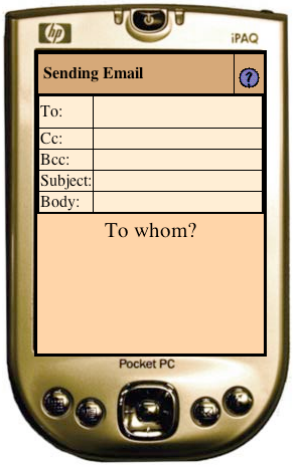


We may choose not to show the contact’s name, even if we have the information. This will reduce the cognitive load. However, the information is recorded as being un realized.



B.2 Full Example

Here is an example (B.1) of the output for a request for a user name. In this scenario, the user has already asked to send an email.



(a)

Figure B.1: Voice: “Who should be on the to list?”

The input to the MUG for this example is as follows:

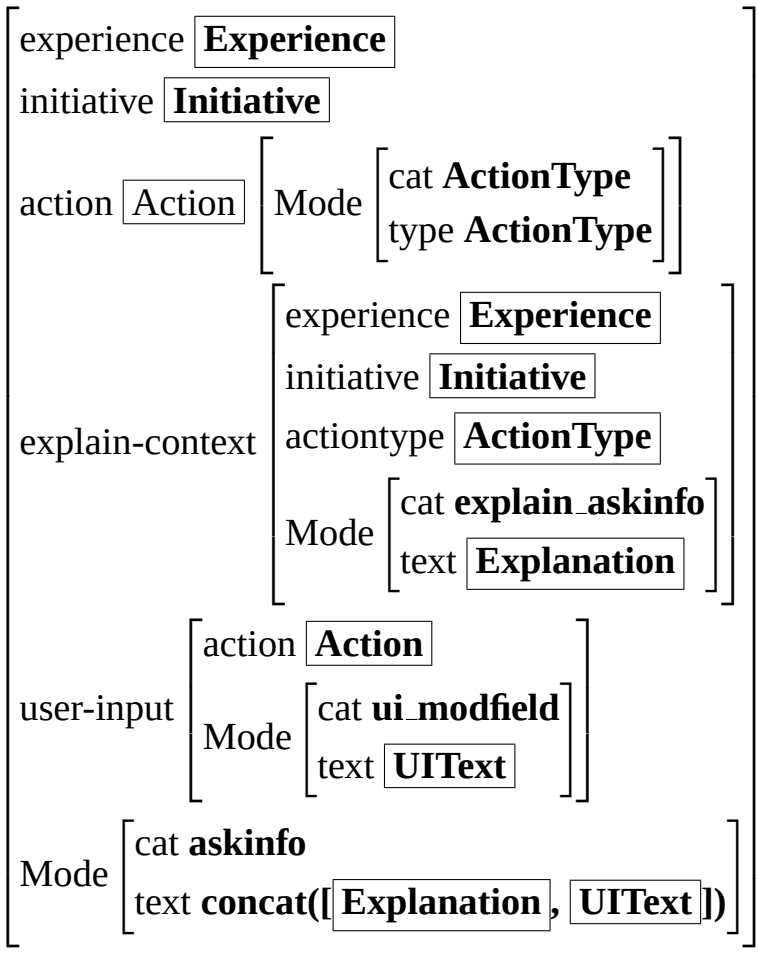
$$\begin{bmatrix} \text{type } \textbf{askinfo} \\ \text{experience } \textbf{expert} \\ \text{error } \textbf{none} \\ \text{action} \begin{bmatrix} \textbf{type addtolist} \\ \textbf{task} \begin{bmatrix} \textbf{type send-email} \\ \textbf{contexttype email} \\ \textbf{email} \left[\textbf{type email} \right] \end{bmatrix} \\ \textbf{scope} \langle \textbf{task/email/to} \rangle \end{bmatrix} \end{bmatrix}$$

That is, we are asking for information from a user who has used the system in the past (expert). They will be adding to a list (task/email/to), and there is no current email message.

Following are all of the components used to generate the screen_dynamic portion of the display.

$$\begin{bmatrix} \text{tree} & \begin{bmatrix} \text{action} & \begin{bmatrix} \text{scope} & \boxed{\textbf{Scope}} \\ \text{task} & \begin{bmatrix} \text{scope} & \boxed{\textbf{Scope}} \end{bmatrix} \end{bmatrix} \\ \text{type} & \boxed{\textbf{DialogueAct}} \\ \text{screen\_dynamic} & \begin{bmatrix} \text{cat} & \boxed{\textbf{DialogueAct}} \\ \text{text} & \boxed{\textbf{A}} \\ \text{type} & \boxed{\textbf{DialogueAct}} \end{bmatrix} \end{bmatrix} \\ \text{screen\_dynamic} & \begin{bmatrix} \text{cat} & \textbf{multimodal} \\ \text{text} & \textbf{template([`}\langle\ \textbf{table}\rangle\langle\textbf{tr}\ \rangle\langle\ \textbf{td}\ \rangle\ \sim\textbf{w}\rangle\textbf{tr}\rangle\langle\textbf{td}\rangle\langle\textbf{table}\rangle\textbf{', [[}\boxed{\textbf{A}}\textbf{]]])} \end{bmatrix} \end{bmatrix}$$

The component above calls for something that is cat:DialogueAct. Unifying that with the input FD, we see that the DialogueAct is askinfo. The component below unifies with that.



askinfo_1 works for either screen_dynamic or voice. It calls for the concatenation of a context explanation and a user input.

$$\left[\begin{array}{l}\text{scope } \boxed{\textbf{Fields}} \\ \text{user\_intention } \boxed{\textbf{User\_Intention}} \\ \text{fieldcontent } \left[\begin{array}{l}\text{scope } \boxed{\textbf{Fields}} \\ \text{content\_source } \boxed{\textbf{User\_Intention}} \\ \text{Mode } \left[\begin{array}{l}\text{cat } \textbf{fieldcontent} \\ \text{text } \boxed{\textbf{Content}} \\ \text{realized } \textbf{1}\end{array}\right]\end{array}\right] \\ \text{fieldtext } \left[\begin{array}{l}\text{scope } \boxed{\textbf{Fields}} \\ \text{Mode } \left[\begin{array}{l}\text{cat } \textbf{fieldtext} \\ \text{text } \boxed{\textbf{FieldText}} \\ \text{realized } \textbf{1}\end{array}\right]\end{array}\right] \\ \text{Mode } \left[\text{text } \textbf{template([`add } \sim \textbf{ to the } \sim \textbf{ w', [}\boxed{\textbf{Content}}\textbf{, }\boxed{\textbf{FieldText}}\textbf{])}\right]\end{array}\right]$$

$$\left[\begin{array}{l}\text{experience } \textbf{expert} \\ \text{Mode } \left[\begin{array}{l}\text{text } \textbf{“”} \\ \text{realized } \textbf{0}\end{array}\right]\end{array}\right]$$

In this example, for the explain_askinfo component, this user is an expert, so there is no explanation.

$$\left[\begin{array}{l}\text{scope } \left\langle \textbf{ task/ } \boxed{\textbf{?}}\textbf{/ } \boxed{\textbf{?}} \right\rangle \\ \text{Mode } \left[\begin{array}{l}\text{text } \textbf{“”} \\ \text{realized } \textbf{1}\end{array}\right]\end{array}\right]$$

The fieldcontent returns nothing.

$$\left[\begin{array}{l}\text{scope } \left\langle \boxed{\textbf{?}}\textbf{/email/to} \right\rangle \\ \text{Mode } \left[\begin{array}{l}\text{text } \textbf{“to list”} \\ \text{realized } \textbf{1}\end{array}\right]\end{array}\right]$$

The fieldtext returns “to list”.

$$\left[\begin{array}{l}\text{action}\ \left[\begin{array}{l}\textbf{type addtolist}\\ \textbf{scope}\ \langle\ \textbf{task/email/to}\ \rangle\\ \textbf{Mode}\ \left[\textbf{realized 1}\right]\end{array}\right]\textbf{Mode}\ \left[\begin{array}{l}\textbf{text “To whom?”}\\ \textbf{realized 1}\end{array}\right]\end{array}\right]$$

The ui_modfield component returns “To whom?”

More grammar details will be available in deliverable D5.4, the Adaptive multi-modal fission/fusion service.

Bibliography

André, E., Finkler, W., Graf, W., Rist, T., Schauder, A. & Wahlster, W. (1993). Wip: The automatic synthesis of multimodal presentations, *in* M. T. Maybury (ed.), *Intelligent Multimedia Interfaces*, AAAI Press, Menlo Park, CA, pp. 75–93.

Beringer, N., Kartal, U., Louka, K., Schiel, F. & Türk, U. (2002). PROMISE - a procedure for multimodal interactive system evaluation, *Proceedings of the LREC 2002 Workshop on Multimodal Resources and Multimodal Systems Evaluation*, Athens, Greece.

Bolt, R. A. (1980). Put-that-there:voice and gesture at the graphics interface, *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, Seattle, pp. 262 – 270.

Bontcheva, K. (2003). Reuse and challenges in the evaluation of NLG systems, *Proceedings of the EACL-2003 Workshop on Evaluation Initiatives*, Budapest, Hungary.

Cassell, J., Sullivan, J., Prevost, S. & Churchill, E. (2000). *Embodied Conversational Agents*, MIT Press, Cambridge, MA.

Cockton, G., Lavery, D. & Woolrych, A. (2002). Inspection-based evaluations, *in* J. Jacko & A. Sears (eds), *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications*, Lawrence Erbaum Associates, Mahwah, NJ.

Coutaz, J., Salber, D. & Carraux, E. (1996). Neimo, a multimodal usability lab for observing and analyzing multimodal interaction.

Elhadad, M. & Robin, J. (1992). Controlling content realization with functional unification grammar, *in* R. Dale, E. Hovy, D. Roesner & O. Stock (eds), *Pro-*

ceedings of the Sixth International Workshop on Natural Language Generation, Springer Verlag. Lecture Notes in Artificial Intelligence, pp. 89–104.

Feiner, S. & McKeown, K. (1990). Coordinating text and graphics in explanation generation, *Proc. of AAAI-90*, Boston, MA, pp. 442–449.

Feiner, S. K. & McKeown, K. R. (1998). Automating the generation of coordinated multimedia explanations, *in* M. T. Maybury & W. Wahlster (eds), *Intelligent User Interfaces*, Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Gajos, K. & Weld, D. S. (2004). Supple: Automatically generating user interfaces, *Proceedings of IUI-2004*, Funchal, Portugal.

Grice, H. (1975). Logic and conversation, *in* P. Cole & J. Morgan (eds), *Syntax and Semantics*, Vol. 3, Academic Press, pp. 41–58.

Grosz, B. J., Joshi, A. K. & Weinstein, S. (1995). Centering: A framework for modeling the local coherence of discourse, *Computational Linguistics* **21**(2): 203–225.

Hart, S. G. & Staveland, L. E. (1988). Development of a multi-dimensional workload rating scale: Results of empirical and theoretical research, *in* P. A. Hancock & N. Meshkati (eds), *Human mental workload*, Elsevier, Amsterdam, The Netherlands.

Johnston, M. (1998). Unification-based multimodal parsing, *Proceedings of COLING-ACL-1998*, pp. 624–630.

Johnston, M. & Bangalore, S. (2000). Finite-state methods for multimodal parsing and integration, AT&T Labs - Research.

Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S. & Maloor, P. (2002). Match: An architecture for multimodal dialogue systems, *Proceedings of ACL-2002*.

Kay, M. (1979). Functional grammar, *Proceedings of the Fifth Meeting of the Berkeley Linguistics Society*, Berkeley, CA, pp. 142–158.

Kieras, D. (2002). Model-based evaluations, *in* J. Jacko & A. Sears (eds), *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications*, Lawrence Erbaum Associates, Mahwah, NJ.

Klemmer, S. R., Sinha, A. K., Chen, J., Landay, J. A., Aboobaker, N. & Wang, A. (2000). Suede: A wizard of oz prototyping tool for speech user interfaces, *CHI Letters: Proceedings ACM Symposium on User Interface Software and Technology*, pp. 1–10.

Knight, J. F., Baber, C., Schwirtz, A. & Bristow, H. W. (2002). The comfort assessment of wearable computers, *Proceedings of the Sixth International Symposium of Wearable Computers*, Seattle, Washington.

Maybury, M. T. & Wahlster, W. (eds) (1998). *Intelligent User Interfaces*, Morgan Kaufmann Publishers, Inc., San Francisco, CA.

McInnes, F., Jack, M., Carraro, F. & Forster, J. (1997). User responses to prompt wording styles in a banking service with a wizard of oz simulation of word-spotting, *Proceedings of IEE Colloquium on Advances in Interactive Voice Technologies for Telecommunications Services*, pp. 1–6.

McNeill, D. (1992). *Hand and mind: What gestures reveal about thought*, University of Chicago Press.

Oviatt, S. (1999). Ten myths of multimodal interaction, *Communications of the ACM* **42**(11): 74–81.

Oviatt, S., Cohen, P., Fong, M. & Frank, M. (1992). A rapid semi-automatic simulation technique for investigating interactive speech and handwriting, *Proceedings of the International Conference on Spoken Language Processing 2*, pp. 1351–1354.

Oviatt, S., DeAngeli, A. & Kuhn, K. (1997). Integration and synchronization of input modes during multimodal human-computer interaction, *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, pp. 415–422.

Panttaja, E., Reitter, D. & Cummins, F. (to appear 2004). The evaluation of adaptable multimodal system outputs, *Proceedings of the Workshop on Multilingual Linguistic Resources (MLR2004), at COLING*.

Pirhonen, A., Brewster, S. A. & Holguin, C. (2002). Gestural and audio metaphors as a means of control for mobile devices, *Proceedings of ACM CHI2002*, ACM Press, Addison-Wesley, Minneapolis, Minnesota, USA.

Reiter, E. & Dale, R. (2000). *Building Natural Language Generation Systems*, Cambridge University Press.

Reitter, D., Panttaja, E. & Cummins, F. (2004). UI on the fly: Generating a multimodal user interface, *Proceedings of Human Language Technology conference 2004 / North American chapter of the Association for Computational Linguistics (HLT/NAACL-04)*.

Roth, S. F. & Hefley, W. E. (1993). Intelligent multimedia presentation systems: Research and principles, *in* M. T. Maybury (ed.), *Intelligent Multimedia Interfaces*, AAAI Press, Menlo Park, CA.

Salmon-Alt, S. & Romary, L. (2000). Generating referring expressions in multimodal contexts, Proceedings of the INLG.

Turk, U. (2001). The technical processing in smartkom data collection: a case study, *Proceedings of Eurospeech2001*, pp. 1351–1354.

Wahlster, W. (2002). Smartkom: Fusion and fission of speech, gestures, and facial expressions, *Proceedings of the 1st International Workshop on Man-Machine Symbiotic Systems*, Kyoto, Japan.

Walker, M., Joshi, A. & Prince, E. (1997a). Centering in naturally occurring discourse: An overview, *in* M. A. Walker, A. K. Joshi & E. Prince (eds), *Centering Theory in Discourse*, Oxford University Press, Oxford, pp. 1–28.

Walker, M., Litman, D., Kamm, C. & Abella, A. (1997b). PARADISE: A framework for evaluating spoken dialogue agents, *in* P. R. Cohen & W. Wahlster (eds), *Proceedings of the ACL-EACL-1997*, Association for Computational Linguistics, Somerset, New Jersey, pp. 271–280.

Woods, D. & Roth, E. (1988). Cognitive systems engineering, *in* M. Helander (ed.), *Handbook of Human-Computer Interaction*, Elsevier, North Holland, pp. 1–43.

Wyard, P. & Churcher, G. (1998). A realistic wizard of oz simulation of a multimodal language system, *Proceedings of 5th. International Conference on Spoken Language Processing*, pp. 1351–1354.

Yang, Y., Okamoto, M. & Ishida, T. (2000). Applying wizard of oz method to learning interface agent, *IEICE Transactions Fundamentals* **E00-A**(1): 1–8.