
Fundamentals of Dialogue Systems

2.1 Introduction

This chapter presents an introduction to the fundamental concepts and developments that shaped the field of dialogue systems research, and thus presents the basis of our work. Speech act theory, together with the plan-based notion of rational agency, has been the predominant framework of most theoretical approaches to both human-human and human-computer dialogue and is still highly influential today. However, conversely to this process that may be called “top down,” as it starts at the high level of intentions, in the dialogue systems community a “bottom up” movement has become popular. It can be characterised as aiming to build actual systems, especially using speech, that can realise specific dialogues at the surface level without a major theoretical or computational overhead. This has led to the development of dialogue scripting approaches, such as VoiceXML, among others.

We see our work as a building block in a compromise between the two directions. Although for our approach we do not adopt a plan-based framework which in variation is the basis of much of the research described this chapter, we recognise that the plan-based approach has led to important notions and modelling approaches, such as the separation of the domain level from other levels of representation in the discourse. Our approach to dialogue management, instead, is based on two later developments: VoiceXML and information state-based approaches.

The chapter is organised as follows: In the following we will give a brief overview of the development of the plan-based approach to analysing discourses and dialogues, before discussing some of the developments in more detail in Sections 2.2, 2.3, and 2.4. Sections 2.5 and 2.6 are dedicated to the converse trends of dialogue scripting languages and information state-based approaches. Finally, we will discuss a selection of implemented systems that exhibit the features relevant to our work in Section 2.8. On this basis, we will draw conclusions as to our architecture and design decisions.

Trying to understand and formalise how humans use language in order to achieve goals has a long tradition in linguistics and philosophy. Rather than isolated utterances most analyses address *discourses*, i.e. ordered sequences of utterances (or conversational acts) that aim to achieve a common purpose in the audience. The goal of modelling discourse is to understand how the individual parts of the discourse contribute to the overall interpretation of the discourse. A discourse can be written or oral, and it can be a monologue, or a dialogue between two or more discourse participants. Although a body of letters or e-mail messages can constitute a discourse, most attention is usually paid to face-to-face discourses.

A *dialogue* can be regarded as a discourse that involves more than one active participant, in the sense that each participant contributes to the discourse. A dialogue is an exchange of messages (utterances) between these *dialogue participants*. In a dialogue, a sequence of consecutive utterances by one dialogue participant is grouped into a *turn*. *Turn taking* refers to one participant assuming or yielding the turn in a dialogue.

Much of today's theoretical understanding of conversational acts is based on the works of Austin [1962] and Searle [1969]. They introduced the notion of a *speech act* which recognises the fact that speaking is a form of *acting* in the sense that the speaker wants to achieve some effect in the mental state of the hearer. Examples of speech acts include asserting, questioning, and commanding.

Grice [1975] introduced the notion of *conversational maxims* as a realisation of a cooperativity principle in dialogue. These maxims constitute rules that are employed commonly (and sometimes without conscious thought) in order to interpret utterances. The basic idea is that using the rules the hearer of a message can draw inferences (*conversational implicatures*) about the speaker's intentions behind an utterance. These inferences allow the hearer to further interpret an utterance beyond the literal meaning of the words spoken. Grice distinguishes between the maxims of quantity, quality, relevance, and manner. For instance, the maxim of relevance would allow the hearer to infer that the speaker considers the information "it is cold here" to be actually relevant in the discourse and that he wants to achieve some effect in the hearer by uttering it.

Building upon Searle's work plan-based models have been developed that view speech acts as actions within a more general plan representing the intentions of the speaker [19, 20]. Formalising speech acts in a plan-based framework allows to apply and adapt the achievements in general planning research to the problem of discourse analysis. The framework also furthers the formalisation by requiring applicability conditions (preconditions) and effects (postconditions) to be specified for speech acts, which may then be regarded as *planning operators*. A further advantage of the plan-based framework is that it may be the basis for understanding how conversational actions are intertwined with physical non-communicative actions. Cohen and Perrault concentrate on the formalisation of REQUEST and INFORM acts. In doing so, questions can be

reduced to requesting to be informed. They formalise the planning operators in a modal logic framework with a possible worlds semantics. Cohen and Perreault present a language for describing planning operators and states of the world. They do not focus on how the speech acts can be realised using words or how a hearer can recognise speech acts from an utterance.

Grosz and Sidner [1986] present a theory of discourse that introduces the distinction of three interrelated structures, namely the *linguistic structure*, the *intentional structure*, and the *attentional structure*. This distinction serves to separate the intentional structure from the other aspects of the discourse and thus allows to develop specific models for it. The distinction has since been adopted by a large body of subsequent work.

Refining the notion of plans as complex mental attitudes [22], Grosz and Kraus [1993, 1996] develop the notion of a *SharedPlan*. A *SharedPlan* is a plan developed by a group of agents in order to achieve a shared goal. The goal is to [24, p.1]

[...] provide a basis for constructing computer agents that are fully collaborative as well as to provide a framework for modelling the intentional component of dialogue.

SharedPlans aim to explain how group activities can be decomposed to the level of individual plans and actions. The model defines how agents can identify ways to perform a group action (i.e. find a *recipe* and identify the necessary parameter values) and how to avoid performing conflicting actions. Lambert [1991] introduces a tripartite model of discourse that emphasises the distinction between the communicative level, the problem solving level and the domain level. This distinction is relevant to our work. Essentially, our proposed reasoning engine applies to the domain level, while processes on the problem solving and discourse levels are coordinated by the dialogue manager. We will discuss some of the developments in more detail in the next sections, before moving on to describe actual system implementations.

2.2 The Structure of Discourse

This section reviews fundamental work concerning how to model different aspects of discourse. Grosz and Sidner's theory of discourse [21] is based on the distinction of three interrelated structures. Each of these captures a different aspect of the utterances that constitute the discourse. The utterances may be in written or in oral form and the discourse may involve only one, two, or many conversational participants (CPs). The theory is generic in the sense that it does not presuppose a human-human or a human-computer discourse. It tries to be applicable to all kinds of discourses. Thus, the aim of the work is to provide insights for psychologists analysing human discourses, as well as researchers and engineers involved in constructing human-machine dialogue systems.

The three different constituents of the discourse model are described in the following: The *linguistic structure* contains the sequence and structural relations of the utterances of the discourse; it aims to represent the “natural aggregation” of utterances into *discourse segments*. The kinds of aspects that are dealt with in the linguistic structure are related to linguistic phenomena, such as anaphora resolution (referring expressions), question-answer pairs and the like. The aggregation of utterances into discourse segments is analogous to the aggregation of words in an utterance, with each component playing a specific role in the entire structure. Discourse segmentation is not solely based on the order of the utterances in the discourse. Non-consecutive utterances may be in the same segment. Certain psychological and psycholinguistic effects (such as differences in pauses lengths) provide evidence for discourse segments. In addition, cue phrases and prosodic variations may also act as markers and are indications of discourse segment boundaries. The authors argue that these markers can be distinguished as to whether they explicitly influence the intentional or the attentional structure. The structural relations between discourse segments reflect the relationships among components of the intentional structure. This structure is discussed in the following.

The *intentional structure* deals with the “discourse-relevant purposes” that are part of the discourse. These kinds of purposes and the structure of these purposes are related to the coherence of the dialogue and also to the question if a sequence of utterances constitutes one or many discourses. For instance, a discourse that serves the single goal of convincing an audience and in which this purpose is pursued in a structured manner is usually regarded as very coherent. Although a discourse participant may have more than one purpose in participating (for instance, to entertain, as well as to describe some event), one purpose of the discourse is seen as foundational and is called the *discourse purpose* (DP). This intention provides the reason for the speaker conveying the particular content of this discourse. At a finer level of detail, each of the discourse segments is associated with a *discourse segment purpose* (DSP) which describes how the discourse segment contributes to the overall goal of the discourse (the DP). Discourse purposes are intentions that are to be recognised by the audience (rather than being private). In fact, some of their effects are not achieved unless the intentions are recognised. Compliments, for instance, are intended to be recognised as such. The authors distinguish the following types of intentions that are candidates for serving as DPs or DSPs: These intentions express a “want” that some other agent

- perform a physical act,
- believe some fact, or believe that some fact supports another,
- intend to identify some object, or
- know some property of an object.

The two major structural relations that link discourse purposes in a hierarchical fashion are dominance and satisfaction precedence. A discourse segment purpose dominates another if the latter contributes to the first (i.e. the latter

is part of a plan to achieve the first). Satisfaction precedence introduces a kind of temporal ordering that specifies that certain discourse segment purposes have to be satisfied before others.

The third element of the discourse model is the *attentional state*. It represents the discourse participants' focus of attention and contains information about the salient objects, their properties and relations. It is a dynamic structure that evolves in parallel with the other component structures. It may change more substantially while, for instance, the linguistic structure is growing in a monotonic fashion. The attentional state is regarded a property of the discourse, and not one of the discourse participants. The attentional state is organised as a set of so-called *focus spaces*, the focusing structure. These focus spaces can be added and deleted in an update process that is governed by so-called transition rules. The whole process is referred to as *focusing*. Each focus space includes the entities that are salient in a particular discourse segment, as well as the discourse segment purpose. Focusing associated the focus space with the discourse segment.

The different constituents contain the information necessary for the conversational participants to decide how to interpret a new utterance, i.e. how it fits into and modifies the given discourse structure. The authors note the importance to distinguish between the three constituent structures for explaining certain discourse phenomena. The authors also argue that their theory of discourse is not a theory of discourse meaning. They point out, however, that such a theory would most likely include a theory of discourse structure as a building block.

2.3 A Hierarchical Model of Intentions

This section reviews work concerning the distinction of different levels of the representation of actions and intentions in a dialogue. While the distinctions introduced in the previous section may be presented as a horizontal split, the model presented in this section focuses on the intentional structure and splits it vertically (cf. [Figure 2.1](#)). The relevance to our approach stems from the introduction of the notion of *domain level*.

In order to represent user intentions in a plan-based approach to dialogue modelling, Lambert [1991] introduces a distinction between three kinds of actions and intentions: *communicative or discourse actions*, *problem solving actions*, and *domain actions*. This corresponds to a tripartite hierarchy of levels (cf. [Figure 2.1](#)). At the lowest level, the *discourse* or *communicative level*, discourse acts are represented and disambiguated. Discourse acts create and modify goals and acts on the second, intermediate, level, the *problem solving level*. In turn, actions on this level modify the *domain level* (the highest level) where the actual goal of the planning process is specified. On the domain level, goals such as travelling by train are represented. The problem solving level contains actions such as instantiating a parameter in a plan to travel, and the

discourse level handles goals such as obtaining information, clarification, or expressing uncertainty.

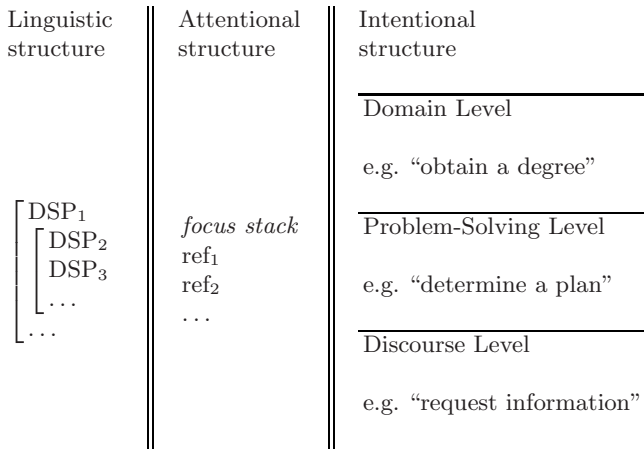


Fig. 2.1. The hierarchical structure of dialogue [25] in the context of the discourse structure proposed by [21].

Actions on the lower levels modify plans on the higher (more abstract) levels. This implies that, when executed, discourse actions can modify the current problem solving plan. For instance, a discourse-level “Surface-Inform” (see below) can define a parameter value in a problem solving-level plan. In turn, completed problem solving actions are used to modify the domain plan. For instance, choosing a particular problem-solving plan may introduce domain-level actions. Conversely, information needs related to modifications on the higher levels initiate actions on the lower levels. For instance, using a particular plan recipe on the problem-solving level may initiate a “Surface-Request” due to open parameter values in the recipe.

The three levels are represented as tree structures, making explicit the hierarchical ordering (and contribution relation) between the actions. The three levels are linked according to the modification/initiation relation described before. Within the tree structures, hierarchical plans (“subaction-arc”) are constructed. The leaves of the discourse-level tree consist of surface realisations such as “Surface-Inform” or “Surface-Request”. These are parts of a more high-level communicative plan. For instance, one may issue a “Surface-Inform” in order to make a subsequent “Surface-Request” acceptable (as required and defined by the discourse-level plan library.) “enable-arc” links connect the discourse level to the problem solving level (as well as the problem solving level to the domain level).

One important insight is how to distinguish between the domain level and the problem solving level: The authors note that the planning agent (e.g. the

user) is the agent of all domain-level actions whereas at the problem-solving level, both participants collaborate as joint agents. Thus, the problem-solving level can be considered a *meta-level* with respect to the domain level. In later work the question of collaborative conflict detection and resolution on the basis of the tripartite structure is addressed [26, 27].

Ramshaw [1991] presents an alternative three-layered model of discourse. However, his focus is on using a middle layer between the domain layer and the discourse layer in order to allow for the exploration of domain plans. For instance, in a banking domain, the exploration of a domain plan that involves opening a banking account may consist of exploring different variants of that plan using different instantiations of parameters. Ramshaw argues that the different layers are necessary to capture different phenomena occurring when the speaker “is on a certain level” or when he switches levels.

2.4 Collaborative Negotiation

This section introduces the concept of collaborative negotiation which is relevant to our work in the sense that it proposes a methodology of how dialogue participants can cooperate to reach agreements concerning beliefs in dialogue.

The notion of collaborative negotiation refers to cooperative behaviour where two or more participants try to establish a common set of beliefs in order to solve a shared problem. In adversarial negotiation, on the other hand, one participant tries to win over another by achieving an agreement that is more advantageous to him than to the other participant. This is ruled out in collaborative interactions. In particular, the absence of deception or intentional misinformation is assumed in collaborative negotiation.

Sidner [1994] has presented an “Artificial Discourse Language for Collaborative Negotiation”. This discourse language aims at modelling the interactions that occur in collaborative negotiation. The language consists of messages to propose, reject, counterpropose beliefs, or seek supporting information for certain beliefs. The term “belief” refers to arbitrary statements in an application domain. These messages are postulated in order to abstract from natural language such as English and to formalise part of the intentional structure in the sense of [21] (cf. Section 2.2). Two dialogue participants, called *agents* in the authors’ terminology, send out messages of different types to each other. After each message is sent, certain logic conditions which are part of the language definition can be assumed to hold. This process is not necessarily monotonic, a belief revision system (or truth maintenance system) is assumed in order to drop certain previous assumptions which are inconsistent with new knowledge. This is necessary, because agents may have revise some of their beliefs. However, the details of that mechanism are not discussed. During the discourse a stack of “open” and “rejected” beliefs is maintained.

We summarise some of the most important message types here:

- **ProposeForAccept:** PFA *agent*₁ *belief agent*₂: This message is sent by *agent*₁ to *agent*₂ in order to propose *belief* as a mutual belief. *agent*₂ should acknowledge the message (which does not imply accepting the belief). The belief is added to the “open” set.
- **AcknowledgeReceipt:** AR *agent*₁ *belief agent*₂: This message is sent by *agent*₁ to *agent*₂ in order to signal that *agent*₁ has received the proposal by *agent*₁. It does not imply that *agent*₁ is accepting the belief. The message is still important because *agent*₂ can now assume that both know as a mutual belief that *agent*₂ holds *belief*.
- **Reject:** RJ *agent*₁ *belief agent*₂: This message is to signal that *agent*₁ does not hold *belief*. *belief* is retracted from the “open” set.
- **AcceptProposal:** AP *agent*₁ *belief agent*₂: Receiving this message, *agent*₂ can infer that both agents share *belief*. Furthermore, the proposed *belief* is retracted from the “open” set.

The beliefs that are communicated are constructed from the operators *BEL* (belief), *INT* (intend), *MB* (mutual belief), and the following primitives: *Communicated()*, *Should-do()*, *Achieve()*, *Supports()*, *Provide-Support()*, *Tell-if()*, *Identify()*, and *Able()*.

Questions and commands can be expressed using the introduced primitives: For instance, a question of the form “Why X?” is analysed as follows:

```
(PFA agent1 (Should-Do agent2
(Provide-Support X context)) agent2)
```

Similar analyses can be assigned to utterances of the form “What is X?” (using *Identify()* instead of *Provide-Support()*), “Can you X?” (*Able()*), or factual questions like “Did John come?” (without specific primitive).

The authors also develop a process model that tries to explain how each message updates the dialogue state. Building on the concept of a stack-based attentional state introduced in [21], stacks for open and rejected beliefs are introduced. However, the author assumes a general form of “automated belief revision system to track all the mutual beliefs” in order to make the process model executable. Sidner’s artificial language was also used as a basis for later information state-based approaches [30, 31].

2.5 Dialogue Scripts and Voice Browsers

In contrast to the traditional, highly generic, approaches previously outlined, this section presents a completely different approach to developing systems that enable a dialogue interaction, especially using spoken language. This section mainly focuses on VoiceXML, but this is by no means the only dialogue scripting language, nor is it the first one. However, it is arguably the most successful one as far as adoption by developers is concerned.

In the late 1990s, the World Wide Web (WWW) was becoming vastly popular and so did hypertext mark-up languages (HTML) for publishing content

in that medium. Soon it was recognised that not only static content was to be published, but in addition interactive applications which in principle rely on the same kinds of user input that usual graphical computer applications need. Hyperlinking was not sufficient to allow for this kind of input, thus the idea of an HTML *form* was born, and – despite its limitations and different incompatible enhancements – has become vastly successful.

It was that development that inspired the growing telephony speech industry to establish a standard for web-based voice applications that should achieve a similar success. The idea was to develop an approach that would allow to quickly deploy basic voice-based applications without a constraining theoretical overhead. Thus, the Voice Extensible Mark-up Language (VoiceXML) 1.0 specification was proposed by a consortium of major telecommunications and software companies (AT&T, Motorola, Lucent, IBM, among others) in 2000. The current W3C recommendation version 2.1 adds certain extensions and improvements on this basis [32]. A simple VoiceXML application is illustrated in Figure 2.2.

```
<vxml>
  <form id="form1">
    <block>Hello user!</block>
    <field name="username">
      <grammar uri="names.srgs" />
      <prompt>What is your name?</prompt>
    </field>
  </form>
</vxml>
```

Fig. 2.2. A simple VoiceXML application.

The most important aspect of the VoiceXML specification is the fact that it defines an abstract processing model for VoiceXML applications, called the *Form Interpretation Algorithm* (FIA). The FIA assumes that initially the system has the turn, and it determines which actions the system should take, in particular how to query for user input, and how to respond to user input or failure conditions (events), if any. The basic structure that the FIA operates on is the *form*. It consists of *items* which belong to different classes, the most common two of which are *fields* and *blocks*. Similarly to forms in HTML, a VoiceXML form consists of smaller units that handle individual form components, i.e. these units allow the user to provide or obtain values of individual task parameters. In HTML, a common element is the `<input>` element, while in VoiceXML, the respective unit is the `<field>`. The VoiceXML approach to form-based dialogues is to iteratively update a dialogue state consisting of a record structure of simple named values. For instance, consider a record containing a departure and destination city, both represented as strings. We

summarise the form processing in the following. In the context of the current form, the FIA goes through the following phases:

1. The *Select phase* determines the next form item by determining which items have not yet obtained a value and are applicable (not prohibited by item guard conditions). If no such item can be determined, the dialogue ends.
2. The *Collect phase* is responsible for playing the item's prompt (or performing executable actions specified in a block). In addition, the speech recognition subsystem is configured to listen for user input and to analyse it with the adequate grammars activated.
3. The *Process phase* processes the value or event obtained from the recognition attempt in the Collect phase. If an exception event occurs because, for instance, the user did not speak or could not be understood, actions defined in a respective event catch handler are executed. Alternatively, if a regular result has been returned, the result is used to fill one or more form items with values. In addition, form handlers which react on filled form items are executed.

Notable similar proposals and approaches include Speech Application Language Tags (SALT) [33] that targets multi-modal applications, and proprietary languages like the Philips Dialogue Specification Language (HDDL) [34], or the Generic Dialogue Modelling Language (GDML) [35] used in embedded systems in the automotive environment.

2.6 Information States and Dialogue Moves

Information state-based approaches present a compromise between the scripting approach described in the last section and the theoretically-inspired plan-based methods. A *dialogue move* is an abstraction similar to speech acts. It represents an action that a dialogue participant performs during the interaction. A *dialogue move engine* (DME, cf. [31]) is a system module that interacts with an information state in a rule-based way in order to integrate observed user moves, and to select a corresponding system move.

The system's basic cycle of operation (its *control algorithm*) consists of the following steps:

1. On the basis of the current information state, try to address obligations by performing and integrating the respective dialogue moves (*update* the information state), then yield turn.
2. Wait for user input by observing the user's dialogue moves and assume turn.
3. Integrate the user's dialogue moves and again update the information state.

Information-state based approaches have been used in a variety of projects. For instance, Matheson et al. [2000] have proposed several mechanisms related to grounding, i.e. the building and management of a common ground of beliefs between dialogue participants. The framework of *Questions Under Discussion* (QUD) has been proposed by Ginzburg [37]. Larsson has refined this framework, and developed an approach called *Issue-Based Dialogue Management* [31]. A sample structure of an information in this approach is shown in Figure 2.3. It may be regarded as a form of typed feature structure [38].

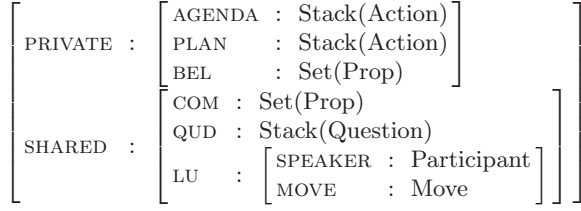


Fig. 2.3. An information state structure, cf. [31, p.36].

An information state represents the beliefs of a system (or participant) at some point in the dialogue interaction. Parts of the information state may be referred to using a path syntax, for instance, /SHARED/LU/MOVE. An information state typically consists of beliefs assumed to be shared (/SHARED) and those which are only held privately by the system (/PRIVATE), either because they have not yet been communicated or because they are only used internally. An information state framework provides the essential base types (for instance, stacks and sets) and operations required to access and operate on the structure.

During the interaction, the information is constantly updated. Typically, this takes place after a user message has been received and after the system has produced its response. The update takes place in terms of dialogue moves and update rules. The most well-known implementation of a dialogue move framework is probably the TrindiKit project [39]. In [40] an alternative formalisation and implementation is proposed.

An example of an update rule as used in TrindiKit is shown in Figure 2.4. When the update rule is applied to an information state, the preconditions specified in the PRE part of the rule are checked. The matching of these conditions may also have the (side-) effect of binding variables to values that can be used in the EFF clause. For instance, this method is used in the update rule to bind the variable Q . Particular languages have been defined to express conditions and updates.

```

RULE : integrateSysAsk
CLASS : integrate
PRE : { $/SHARED/LU/SPEAKER == sys
        $\left\{ in(\$/SHARED/LU/MOVES, ask(Q)) \right.$ 
EFF : {  $push(/SHARED/QUD, Q)$ 

```

Fig. 2.4. An information state update rule, cf. [31, p.43].

2.7 Concepts and Terminology

In the following, some common concepts and terminology from the area of dialogue systems development will be introduced, which will help to better understand the system descriptions in the next section.

Dialogue Initiative. The dialogue participant controlling the dialogue flow is said to have the *dialogue initiative*. Dialogue initiative has to be distinguished from turn taking. A dialogue participant who has assumed the turn may not have the dialogue initiative, for instance, when he assumed the turn just to answer a question raised by another participant.

In dialogue systems the following modes concerning dialogue initiative are distinguished:

- *System initiative* (directive mode): the system is in control of the dialogue and the system's utterances are designed to restrict the user's responses to small set of easily distinguishable options.
- *User initiative* (reactive mode): the user is in control of the dialogue flow, the system merely responds.
- *Mixed initiative*: both system and user may take initiative during the dialogue. The level of system initiative may still vary between *proactive* behaviour, only initiating clarification subdialogues, and a truly negotiated initiative.

Each of the modes has its own advantages and disadvantages depending on the context of use. For instance, novice users may be satisfied with a system-directed dialogue, because that strategy is more robust and guarantees in most cases that progress is made in the dialogue and that the chances for certain types of error such as misrecognitions may be minimised. A similar reason may apply to dialogue under adversary conditions such as noisy environments. Expert users, on the other hand, are more likely to prefer to assume the initiative in the dialogue, since they have a clear idea of what the system can be instructed to do. Technologically, allowing user initiative is more difficult than maintaining system initiative, because the system can establish fewer expectations concerning the next user utterance, which increases the chances of misinterpreting the user.

Dialogue Control. In dialogue systems, the term *dialogue control* refers to how the system's behaviour is implemented technically. Although this is an internal feature of the system, the kind of dialogue control chosen for

the implementation determines to a high degree the kinds of interactions the system can engage in.

- Finite-state models offer a high degree of system control, which is vital in many environments. However, the limited expressiveness of finite-state models often incurs a certain inflexibility of dialogue systems implemented on their basis.
- Frame-based models extend finite-state models with more flexibility, especially concerning some aspects of mixed initiative (e.g. overanswering). A typical example is the VoiceXML form interpretation algorithm.
- Information state-based models are a general way of implementing dialogue strategies which may be used to implement finite-state, frame-based, or more complex models. Moreover, certain it has been used for implementing generic dialogue behaviours and strategies such as grounding and questions under discussion.
- Plan-based or agent-based models have been highly influential approach to modelling interaction and dialogue for a long time. An advantage of a plan-based model is that general behaviour rules may be stated which a dialogue system has to obey. Such rules of behaviour may include cooperation, helpfulness, sincerity. They may be, for instance, based on the Gricean maxims.

However, very strong assumptions regarding the capabilities of the system and the setting have imposed severe limitations on the practical applicability of the framework.

- Statistical models and data-driven methods for dialogues are motivated by the success of these methods in other areas, most notably speech recognition. In principle, the idea of training a dialogue model is appealing. However, the limited availability of training data as well as the uncertainty as to the underlying representation whose parameters were to be trained has prevent widespread use of this approach to date.

2.8 Systems and Architectures

Since we aim at practical dialogue systems development, it is very instructive to review certain implemented SLDS and architectures. With the strong evolution of the field of dialogue systems, the list of systems to be discussed here is necessarily incomplete. For a more detailed presentation of a larger range of systems, we refer to reader to [3]. We focus on those that most clearly exhibit features that are relevant to our work.

Allen et al. [41, 42] introduce a dialogue systems classification scheme according to the following questions:

- Reference capabilities: Which referring expressions can be used in utterances? Possible options are static expressions, i.e. proper names or unique descriptions, and descriptions resolved by status of task. For instance, in

VoiceXML field-level grammars determine such a context. Alternatively, if personal pronouns in general is allowed, a general form of anaphora resolution based on salience and recency is required.

- Task complexity: Does the task consist of only a primitive action or is it composed of (hierarchically) structured actions? Or does the task merely consist of simple queries to a static database (such as in timetable information domain)? In addition, can multiple tasks be handled by the system, or can the task change as dialogue develops? In [43] the following additional questions regarding a plan-based domain are raised: What kinds of plan operations are possible? Is it possible to select and change different the recipe to use, including the temporal order of actions? Can the resources involved be changed? Is it possible to compare different alternative plans?
- Dialogue management capabilities: This concerns the question whether the dialogue itself can be used to manage the interaction. Thus, are certain phenomena and behaviours possible, such as clarification and error correction subdialogues? And if so, who can initiate them? In particular, if they can be only initiated by the system, its capabilities are restricted. Can the user or system or both answer with return questions? (e.g. “Do you mean X?”). Finally, is any form of “meta discussion” possible?
- Speech acts supported: This concerns the kinds of utterances that can be understood by the system. A basic system, for instance, may only produce questions and only accept assertions from the user. However, more complex speech acts such as those introducing obligations or promises are possible.
- Modality: This concerns the available modes to interact with the system. Typical modalities include speech, typed natural language, graphics, input from pointing or writing devices, as well as multimedia presentations on the output side. A key quality is also a system’s ability to fuse input from different modalities and to produce aligned output presentations in different modalities.
- Incrementality: Is it possible to develop topics and utterances incrementally? Also, what forms of back channel responses and grounding are possible?

These questions may serve as a general framework for the discussion of the following systems and architectures.

2.8.1 Circuit Fix-It Shop

A pioneering spoken language dialogue system, called Circuit Fix-It Shop, is presented in [44, 45, 46]. It is relevant to our work because it presents one way to integrate dialogue management with a reasoning engine, in this case, a theorem prover based on Prolog [47].

As one of the earliest working systems, this system contained a speech recogniser with a non-trivial vocabulary. The domain of application is electronic circuit repair, with the system being an expert on the circuit workings,

but without knowledge about concrete repair instances or sensing capabilities. Thus, collaboration and knowledge exchange is crucial in order to solve the common goal of diagnosing and repairing a certain circuit board. According to the authors the system implements essential behaviours required for efficient human-machine dialogue: problem solving to achieve a target, conducting and managing subdialogues, employing a user model, changing initiative during the interaction, and the generation of context-dependent expectations for speech recognition.

From the viewpoint of interaction experience, the system is interesting because it is able to operate in different system modes, termed *directive*, *suggestive*, *declarative*, and *passive*. In the directive mode the system is issuing command-like utterances, and user is expected to follow the instructions of the system. Also, interruptions and transitions to other subdialogues are restricted. The less directive modes provide more flexibility to the user. The system is also able to integrate a user model and evolve it during the interaction. The user model represents system beliefs concerning the knowledge of the user. For instance, whether or not the user knows where a certain knob is on the board. According to [45, p.4],

The role of the user model is thus to supply or fail to supply axioms at the bottom of the proof tree. It both inhibits extraneous verbalism and enables interactions necessary to prove the theorem.

Technologically, the system is Prolog-based, but with certain extensions to the Prolog theorem proving approach in order to enable the interactive construction of a proof. To this end, the so-called *missing axiom theory* is designed to answer the question when and why the system should initiate interaction with the user. The behaviour of the system is driven by a theorem proving procedure that tries to prove a stipulated goal. When the system detects that the proof cannot be completed due to missing information it resorts to interaction with the user as an attempt to gather that information.

Internally, a recursive procedure called *ZmodSubdialog* is responsible for proving a given goal argument in the style of Prolog. Three main cases are distinguished in ZmodSubdialog concerning a goal *R*: If *R* is trivial, i.e. it can be executed by the system in a single step or it can be proved by the user model, the system assumes that this goal has been achieved and the proof proceeds. If *R* is a *vocalize* goal, the goal is achieved by outputting its content as a spoken utterance. Depending on the rule base, the respective goal may require the user to signal understanding and the performance of the indicated command or request. Otherwise, *R* is a complex goal with subactions defined in the knowledge base. In this case, new instantiations of the ZmodSubdialog procedure are created with the appropriate subgoals. More than one instance of a ZmodSubdialog call can be instantiated at one point in time. This represents different subdialogues that can be entered. If a subdialogue is exited, the actions in ZmodSubdialog are suspended, but may be resumed at a later

stage in the interaction. The dialogue control module decides with which subdialogue to proceed.

In the case of a vocalize goal, the user can respond with different replies. In the case of an affirmative response (“Okay”, “I’ve done it.” etc.), the respective goal is proven and the proof procedure can continue. Otherwise, if the user indicates failure to perform or understand the action, the respective proof goal and dependent parts of the proof fail. This usually initiates backtracking in the proof search. Alternatively, the user can initiate clarification subdialogues: The system handles requests for clarification by “dynamically modifying the rule and re-executing with a newly required clarification subdialog.” Finally, if the user responds in a way that is not expected by the current subdialogue, a transition to a different subdialogue is possible. Control is passed to other subdialogue if the user utterance is adequate to that subdialogue’s current state.

Another interesting aspect of the system is the modelling employed in the knowledge base: it contains Prolog rules which are classified in three different categories: *General debugging rules* describe procedures in the circuit repair domain, for instance, how to “set a knob to a value”. Typically, these rules include descriptions of higher-level goals in terms of subgoals. *General dialogue rules* represent the system knowledge about what can be achieved with vocalize goals, and, in particular, which goals can be vocalised. Finally, the *user model rules* describe the system’s beliefs about what the user knows and what actions the user is capable of performing.

2.8.2 TRAINS and TRIPS

The University of Rochester’s TRAINS [48, 49] project and its successor TRIPS [11] are efforts to build interactive planning systems. They feature spoken language interaction in application domains such as logistics, evacuation and emergency planning, and personal assistance, such as medication advice.

In order to develop the TRAINS system, human-human dialogues were recorded that feature rich problem-solving interactions in a logistics domain [50]. Typical problems included building transportation plans to move a certain amount of commodities to a specific location under time constraints. Although the domain is limited to a toy world as far as the number of locations, for instance, is concerned, the kinds of complex problem-solving phenomena that became apparent were highly complex. The actual TRAINS system contained a more limited domain in order to remain implementable. The modelling of the TRAINS domain and aspects of reasoning therein will be described in more detail in Chapters 4 and 5.

Concerning the interaction with the system, the user and system assume asymmetric roles. The user is responsible for the overall scheduling of actions, whereas the system should be supportive [51, 52]. In particular, the system may fill in details (that do not require the user’s attention), suggest solutions,

present and describe the current state of the world and how the current plan may affect it, and dispatch plans to external *agents*. These agents represent the system's facilities to interact with the physical world. The system needs to interpret reports from these agents, and, if necessary, coordinate the correction and modification of plans.

For its internal processing the TRAINS system is based on so-called *Episodic Logic* (EL) [53, 54]. This highly expressive logic is modelled to be closely related to natural language while at the same time being able to serve a practical reasoning tool. For instance, EL includes features like generalised quantifiers, lambda abstraction, sentence and predicate nominalisation, intensional predicates (for expressing wanting, needing, believing etc.), and under-specified representations (for instance, for reference resolution). The main uses of EL within the system are the representation of natural language utterances as obtained by a natural language parser, the modelling of the conversational interaction, and the representation of domain-level information in the form of event-based temporal logic. For these purposes and the processing of these representations, the RHET knowledge representation system has been applied.

TRAINS' successor TRIPS, for The Rochester Interactive Planning System, is a system that substantially generalises and extends the TRAINS approach. TRIPS has been applied to different domains, such as evacuation and medical assistance planning and medication advice, as well as in the military domain [55]. To this end, the TRIPS architecture incorporates a module called *Problem Solving Manager* which can be adapted to different domains. It is responsible for the domain-level problem solving operations such as building and evaluating plans. In the Pacifica evacuation domain, for instance, a human evacuation planning expert and the system collaborate to construct a complex transportation plan that has to fulfil various temporal and resource-dependent constraints. The architecture proposed by [56] is a refined version of the TRIPS architecture and is illustrated in [Figure 2.5](#). It aims to address shortcomings of a so-called "pipelined" architectures in which the processing of user input essentially occurs in a serial fashion. Such processing implies that much time is spent waiting and, perhaps more severely, many translations between representations has to take place. According to the authors, this property limits the usefulness of conventional architectures for mixed initiative. In their proposed architecture, on the other hand, both the user and the system are explicitly treated as agents.

In the architecture proposed the main components, the Interpretation Manager, the Collaborative Agent, and the Generation Manager, operate asynchronously. For instance, the architecture provides the possibility for incremental interactions by enabling the interleaving of utterances with acknowledgement without assuming the user.

Concerning the interaction with the TRIPS system, certain interesting functionalities have been realised, most notably the detection of conflicts in planned activities and the exploration of scenarios using hypothetical reasoning. Domain-level conflicts may occur, for instance, when a bridge used

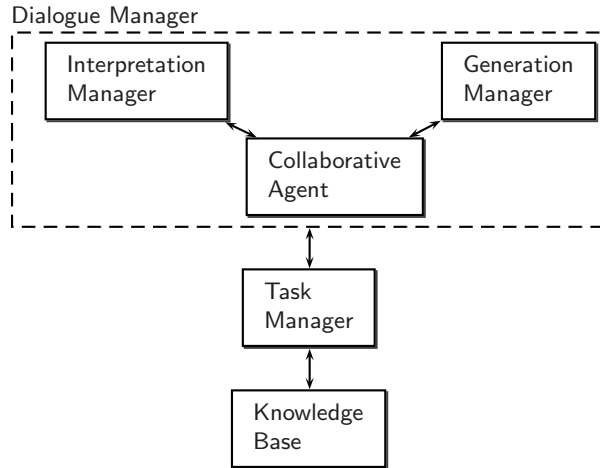


Fig. 2.5. Collaborative SLDS architecture proposed by [57, 56].

in an operation becomes unavailable. In that case, the planned actions affected, transportation events, are marked to indicate that they “may need to be revised”. However, even though conflicts in certain actions are detected, it seems the user does not obtain a significant amount of support from the system concerning the decisions how to resolve the conflict.

Hypothetical reasoning is illustrated in the following slightly adapted dialogue fragment from the TRIPS system in the evacuation domain:

U > What if we went along the coast, instead?
 S > That option would take 10 hours and 42 minutes.
 U > Forget it.

In this dialogue, the user enters a hypothetical reasoning interaction (signalled by the key word “instead”) in order to evaluate a specific scenario that differs from the current plan. This leads a new (and increased) system estimate for the overall time required to complete the planned activities. Consequently, the user cancels the exploration of this scenario.

Regarding the domain level on which these interactions are based, the TRIPS system is able to handle substantially more complex tasks than the TRAINS system. In the Pacifica domain, for instance, different types of transportation methods, each with specific constraints, are modelled. For instance, instead of ground transportation, helicopters may be used, but these are more limited in terms of capacity.

Interestingly, and in contrast to the earlier TRAINS system, the information about planned activities seems to be gathered mainly by performing stochastic simulation. This has advantages and disadvantages. On the one

hand, it can be more realistic and to some extent more expressive than, for instance, a logic-based representation. The simulation of a plan may yield a probability distribution for certain parameter values like the duration. On the other hand, simulation limits the declarative notion of the modelling in the sense that the reasons for certain consequences, as in the dialogue example above, may not be traced back to the fundamental assumptions that implied it (i.e. that the longer duration is caused by the alternative route). Furthermore, the reliance on simulation may limit the ability of the system to deal with partial information because it may not be sufficient to perform a simulation.

2.8.3 COLLAGEN

COLLAGEN [58, 59, 60] is an application-independent *collaboration manager* based on the SharedPlan theory of discourse [61, 24]. The collaboration manager's task consists of assisting a human user in working with a domain application on a specific task, for instance, sending e-mail messages, programming a VCR, or planning a flight itinerary. COLLAGEN uses a discourse interpretation algorithm based on plan recognition [62, 63] in order to match user actions to instances of domain-specific recipes and integrate them into a shared interaction history. The dialogue is modelled in terms of Sidner's artificial discourse language (cf. Section 2.4). This theory is based on the *ProposeForAccept* and *AcceptProposal* primitives.

According to the authors, a key benefit of the domain-independent collaboration manager is the automatic construction of an interaction history that is hierarchically structured according to the user's and agent's goals and intentions. The goal of the collaboration manager in COLLAGEN is to achieve user success in working with some domain application that is accessible through its own direct-manipulation graphical user interface. The user is assisted by the COLLAGEN's *interface agent*, a software module that incorporates some knowledge about the application. Both the user and the interface agent are allowed to work directly with the application user interface. They perform actions such as keyboard or mouse input. They may, for instance, press one of the application's button controls or type in some text into a text field. These actions are observable to the respective other participant. The collaboration manager maintains and updates a plan structure that is derived from the recognised user goals based on a library of domain-specific plan templates, or *recipes*. The plan instance serves as a basis for the generation of suggestions and questions to be presented to the user.

In terms of the approach, COLLAGEN is substantially different from the systems discussed so far. These assumed a central dialogue management component to be in control of the interaction. In particular, it does not provide a general natural language input channel to the user. Instead, a menu-based facility to select possible next actions is proposed. In the COLLAGEN approach, the user is free to ignore the assistance offered by the system and

work directly with the domain application. Also, COLLAGEN does not offer an alternative to replace the interface to the domain application functionality. Instead, it aims at assisting the user in interacting with the existing interface. As such, it bears some resemblance to a tutoring system.

According to Rich and Sidner, the development of an interface assistant for a different application requires only a relatively small additional amount of programming work. The work mainly relies in implementing a COLLAGEN-specific interface layer that translates between recipe steps and concrete application actions such as calling an executable function. An extension and re-implementation of the COLLAGEN approach is the DiamondHelp framework [64]. This Java-based framework emphasises the need for a modular and extensible architecture reusable across domains.

2.8.4 SmartKom

SmartKom [65] is interesting because of its approach to coherent multimodal interaction and distributed modular architecture. It distinguishes between several key components, such as discourse modelling, action planning, and the access to devices and services. SmartKom has also pioneered research into deploying an interface technology in substantially different usage environments, including its usage in a mobile context [66, 67].

On the input side, SmartKom realises a process called *media fusion*, such that user input from different modality streams can be tightly coupled. This is useful in particular for a mutual disambiguation of the input streams. For instance, a user utterance of the form “I want to go there” and a synchronous pointing gesture on a map display can be analysed together in order to resolve the referent of the phrase “there”. In addition to these functionalities which the user can consciously take advantage of, the system attempts to recognise signals sent more subconsciously by the user, such as facial expression to indicate pleasure or anger.

In multimodal dialogue systems, utterance presentation is generalised to a coordinated generation of multimedia presentations. In the SmartKom project, an XML-based multimedia presentation language similar to SMIL has been developed and applied. The goal of this formalism is to achieve timed coordinated and efficient multimedia output, consisting, for instance, of avatar movements including gestures and lip movements, list and map presentations, as well as output of synchronous speech synthesised speech.

The component architecture of the SmartKom system is illustrated in [Figure 2.6](#). Apart from the modality-related functionalities already mentioned, the architecture addresses important aspects related to our work. In particular, on the domain level SmartKom introduces a module called *function modelling* which is responsible for communicating with external task-specific functionalities, such as programming a VCR or telephone. The functionality commonly referred to dialogue management is spread via several tightly interacting components. The central role is assumed by the *action planning*

component which determines the system's (inter-)action goal. To this end, it consults several components here subsumed under the label *knowledge services* which deal which include models of context, discourse, and the overall interaction.

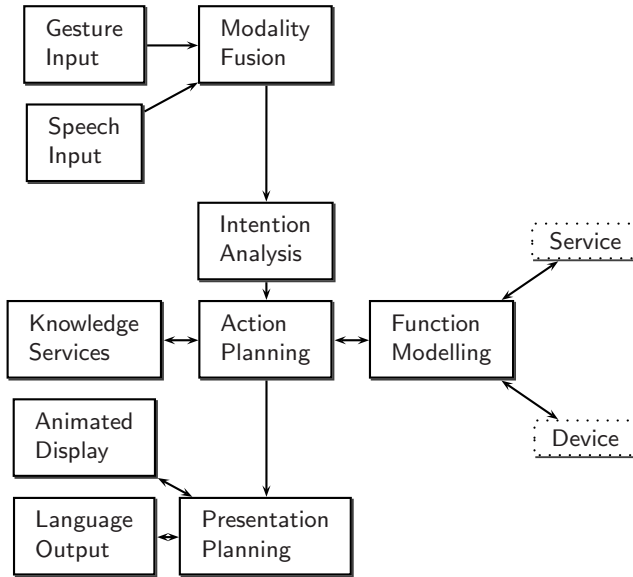


Fig. 2.6. The SmartKom component architecture [66].

The task of the function modelling is to determine which external devices are necessary and how they have to be controlled in order to handle a complex request like “Record the film XYZ tomorrow” in a plug-and-play fashion. The component employs a two-layered functional model consisting of an *internal* and an *external* model. The latter provides information about which functionalities are provided by a device that are relevant externally (to the user). The internal model, on the other hand, includes specialised information encoded in finite state machines that allow the function modelling component to control the device in order to perform specific operations. In this two-layered model, the recording request mentioned above is modelled as involving two devices, the actual VCR and a virtual scheduling device that triggers the action at the specified time.

The function modelling component is clearly related to our goal of integrating application domains. However, in contrast to our approach, the tasks realised within the SmartKom project seem to be tightly focused on the control and operation of external devices.

2.8.5 D’HOMME

Bos and Oka [2002] present an inference-based approach to dialogue system design and its realisation as a prototype in a “smart room” environment. D’Homme is particular in the sense that inference in First-Order Logic plays a central role in both the utterance interpretation and the system’s decision making concerning responses and external actions. As such, the approach is similar to the one presented in [69]. The dialogue modelling is based on Discourse Representation Theory (DRT) [70, 71] which lends itself suitably to a translation into First-Order Logic.

The authors argue for the use of First-Order Logic to capture natural language meaning due to the expressiveness of FOL, its well-understood formalism, and the availability of promising tools. In particular, logical inference can contribute to ambiguity resolution in discourses, for instance, concerning the resolution of anaphoric expressions.

Their inference-based approach is based on the analysis of the complete dialogue, rather than a single utterance. The aim is to find a *consistent* semantic representation capturing the meaning of the dialogue. In particular, the failure to find such an interpretation is considered to signal the presence of problems such as misunderstandings between the dialogue participants.

In the following we will discuss some aspects of the approach in more detail. Consider the following user utterance:

U > Switch every light in the kitchen on!

This is translated by a natural language parser into the discourse representation structure (DRS) which is illustrated in Figure 2.7. In this DRS, the δ operator and the “t:” syntax are examples of the extensions to standard DRT which are used to represent actions and situations.

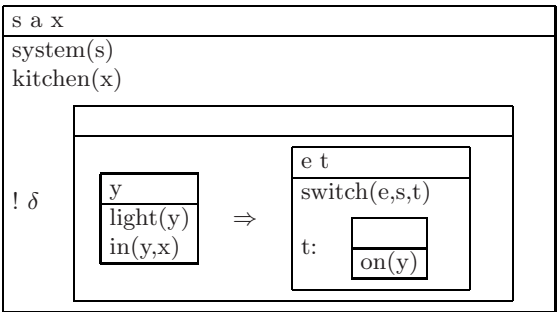


Fig. 2.7. Discourse Representation Structure of an instruction.

The DRS is subsequently translated into a formula of First-Order Logic. One may note that each predicate in the DRS is extended with a possible

world referent. This approach emulates the use of modal logic by a first-order approximation.

$$\begin{aligned} &\exists w, s, a, x . (possible_world(w) \wedge system(w, s) \wedge kitchen(w, x) \wedge \\ &\quad \exists v, a . (action(w, a, v) \wedge \forall y . (light(a, y) \wedge in(a, y, x) \rightarrow \\ &\quad \exists e, t (switch(w, e, s, t) \wedge on(t, y)))))) \end{aligned}$$

The model-theoretic interpretation of this formula does not distinguish between possible worlds and (regular) entities in the domain. Thus, standard first-order inference tools can be applied. The authors also present an update algorithm that defines how complete dialogues can be interpreted using their approach. One of the central decisions that the algorithm has to take concerns the detection of the potential inconsistency of the user input. In the update algorithm, two reasoners are used in parallel. A model generation procedure is used to build models in case the input is satisfiable. At the same time, a theorem prover tries to prove the inconsistency of the problem. Whichever reasoner is first to return a result, will stop the other engine. If the theorem prover is successful, the input will be rejected. If, on the other hand, the input is satisfiable, it will be integrated into the evolving discourse context. In particular, if new entities denoting system actions are inferred, these actions will be executed.

The design presents an innovative approach to a logic-based architecture of dialogue systems. In particular, the flexibility of using a logic-based approach has been shown. However, the following issues present possible areas of improvement. Firstly, the model generation procedure employed is a black box. No proof structures are generated or used. It seems unclear how the system deals with multiple models. In addition to the model generator, a theorem prover for the case of inconsistent specifications is required. So, two different reasoners are used rather than one, which is undesirable. Furthermore, inconsistencies are seen as misunderstandings. However, there may be a lot of other reasons for logical inconsistencies, such as, the specification of an instruction that the system cannot fulfil. Finally, the system does not implement an incremental inference mechanism. This limits its applicability to small dialogues.

2.9 Discussion

This chapter has provided an introduction to some of the fundamental concepts and developments in the field of discourse and dialogue research. Concerning relevance to our approach, the main result of the theoretically motivated plan-based approaches is the definition of a domain level of representation. This level is used for representing the specific tasks that the user wants to accomplish. In such a task-oriented setting, they are the reason for the user to engage in a dialogue either with a human expert or a computer system.

In our approach to dialogue management, we do not follow the tradition built on the plan-based models. These have been criticised for being overly complex and rather abstract in their focus on the underlying intentions between utterances and “neglecting what the dialogue is about,” namely the relation to the actual surface level. The diametrical direction of the dialogue scripting approach focuses on how to build actual systems that can be interacted with by actual users, in particular via speech. This focus stems not least from a commercial interest to provide services in the modality of spoken language. We view our approach based on domain-level reasoning as a building block in a compromise between these directions. We aim at balancing the necessary level of abstraction on the one hand with the desire to implement systems for specific applications on the other hand.

The systems discussed in the previous section show interesting aspects of interaction and dialogue behaviours concerning problem solving in task-oriented domains. However, none of the systems provides a set of functionalities that subsumes our goals of interactive reasoning and domain integration while maintaining a practical level of complexity.

For instance, the Circuit Fix-It system demonstrates how practical problem solving interactions can be modelled with the help of a theorem proving approach, i.e. in a logic-based manner. However, we do not follow this specific direction, since in that approach the theorem proving is applied on the level of the dialogue management, i.e. the different levels of representation concerning the dialogue and the domain are mixed. In our approach the reasoning engine operates exclusively on the domain representation. In particular, we aim at a clear separation of the domain knowledge from the interaction (and dialogue management) knowledge.

The COLLAGEN and DiamondHelp systems present an interesting approach to providing assistance in various application domains, such as travel planning or the control of home appliances. However, to our conception, the approach relies fairly much on plan-based metaphors and constructions. For instance, extensive plan recognition capabilities are required by the system to detect the intentions of the user. In addition, it seems unclear if transparent reasoning and an explanation of the system’s behaviours can be achieved within that framework.

This criticism also applies to some extent to the TRAINS and TRIPS systems, although TRIPS is perhaps the system that pioneered hypothetical reasoning in the sense of “what if” questions in spoken dialogue. The system also provides mechanisms to detect conflicts in planned operations. However, in both cases the information seems to be provided by a simulation module, rather than by inference. In our opinion, this may be useful for obtaining the information, but restricts the ability to reason about the dependencies and thus limits the system’s support for the user to make the right decisions, for instance how to resolve a conflict.

The D’HOMME project illustrates the flexibility that can be achieved by a consequent use of logic formalisms concerning natural language input and

dialogue analysis. For instance, the use of natural language expressions including quantifiers has not been addressed in the other systems discussed here. However, from an architectural perspective the need to integrate two different reasoning engines and their use as autonomous black boxes seem to be candidates for improvement. In particular, in our approach we aim at a tight integration with the reasoning engine that will also be able to provide meaningful proof structures for its inferences. Furthermore, the domains addressed seem to be fairly limited. Thus it is difficult to say if the flexibility provided at the input level is reflected in the domain representation and its processing.

The SmartKom project has also been influential to our approach, also through the personal involvement in the project. From the perspective of domain functionality and usage scenarios, SmartKom has pioneered a movement toward system architectures that can be used in many environments and under different conditions. Also, a large number of services, such as the operation of home appliances, has been implemented on that level. Nevertheless, rather than on integrating these domains on the back-end in the sense that they can contribute to an overall task of the user, the SmartKom's strengths rely in its provision of a multi-modal front-end with novel and powerful features, such as a systematic modality fusion and fission.



<http://www.springer.com/978-1-4419-9727-2>

Domain-Level Reasoning for Spoken Dialogue Systems

Bühler, D.; Minker, W.

2011, XIII, 185 p., Hardcover

ISBN: 978-1-4419-9727-2