A platform for developing Intelligent MultiMedia applications

Tom Brøndsted, Paul Dalsgaard, Lars Bo Larsen,
Michael Manthey, Paul Mc Kevitt, Thomas B. Moeslund,
and Kristian G. Olesen

Technical Report R-98-1004
May, 1998

Center for PersonKommunikation (CPK)
Fredrik Bajers Vej 7A-5
Institute for Electronic Systems
Aalborg University
DK-9220, Aalborg Ø, Denmark
E-mail: cpk@cpk.auc.dk – Fax: (+45) 98 15 15 83 – Tel: (+45) 96 35 86 40

ISSN 0908-1224

A platform for developing Intelligent MultiMedia applications

Tom Brøndsted, Paul Dalsgaard, Lars Bo Larsen,
Michael Manthey, Paul Mc Kevitt[1], Thomas B. Moeslund,
and Kristian G. Olesen
Institute for Electronic Systems (IES)
Fredrik Bajers Vej 7
Aalborg University
DK-9220, Aalborg Ø, Denmark
E-mail: mmui@cpk.auc.dk

[1]Paul Mc Kevitt is also a British Engineering and Physical Sciences Research Council (EPSRC) Advanced Fellow at the University of Sheffield, England for five years under grant B/94/AF/1833 for the Integration of Natural Language, Speech and Vision Processing.

Contents

Abstract

Intelligent MultiMedia (IntelliMedia) focusses on the computer processing and understanding of signal and symbol input from at least speech, text and visual images in terms of semantic representations. We have developed a general suite of tools in the form of a software and hardware platform called *CHAMELEON* that can be tailored to conducting IntelliMedia in various application domains. CHAMELEON has an open distributed processing architecture and currently includes ten agent modules: blackboard, dialogue manager, domain model, gesture recogniser, laser system, microphone array, speech recogniser, speech synthesiser, natural language processor, and a distributed Topsy learner. Most of the modules are programmed in C and C++ and are glued together using the DACS communications system. In effect, the blackboard, dialogue manager and DACS form the kernel of CHAMELEON. Modules can communicate with each other and the blackboard which keeps a record of interactions over time via semantic representations in frames. Inputs to CHAMELEON can include synchronised spoken dialogue and images and outputs include synchronised laser pointing and spoken dialogue. An initial prototype application of CHAMELEON is an *IntelliMedia WorkBench* where a user will be able to ask for information about things (e.g. 2D/3D models, pictures, objects, gadgets, people, or whatever) on a physical table. The current domain is a *Campus Information System* for 2D building plans which provides information about tenants, rooms and routes and can answer questions like "Whose office is this?" and "Show me the route from Paul Mc Kevitt's office to Paul Dalsgaard's office." in real time. CHAMELEON and the IntelliMedia WorkBench are ideal for testing integrated signal and symbol processing of language and vision for the future of SuperinformationhighwayS.

Chapter 1

Introduction

The area of MultiMedia is growing rapidly internationally and it is clear that it has various meanings from various points of view. MultiMedia can be separated into at least two areas: (1) (traditional) MultiMedia and (2) Intelligent MultiMedia (IntelliMedia). The former is the one that people usually think of as being MultiMedia, encompassing the display of text, voice, sound and video/graphics with possibly touch and virtual reality linked in. However, here the computer has little or no understanding of the meaning of what it is presenting.

IntelliMedia, which involves the computer processing and understanding of perceptual signal and symbol input from at least speech, text and visual images, and then reacting to it, is much more complex and involves signal and symbol processing techniques from not just engineering and computer science but also artificial intelligence and cognitive science (Mc Kevitt 1994, 1995/1996, 1997). This is the newest area of MultiMedia research, and has seen an upsurge lately, although one where most universities do not have all the necessary expertise locally. With IntelliMedia systems, people can interact in spoken dialogues with machines, querying about what is being presented and even their gestures and body language can be interpreted.

1.1 Background

Although there has been much success in developing theories, models and systems in the areas of Natural Language Processing (NLP) and Vision Processing (VP) (Partridge 1991, Rich and Knight 1991) there has been little progress in integrating these two subareas of Artificial Intelligence (AI). In the beginning although the general aim of the field was to build integrated language and vision systems, few were, and these two subfields quickly arose. It is not clear why there has not already been much activity in integrating NLP and VP. Is it because of the long-time reductionist trend in science up until the recent emphasis on chaos theory, non-linear systems, and emergent behaviour? Or, is it because the people who have tended to work on NLP tend to be in other Departments, or of a different ilk, from those who have worked on VP? Dennett (1991, p. 57-58) says "Surely a major source

of the widespread skepticism about "machine understanding" of natural language is that such systems almost never avail themselves of anything like a visual workspace in which to parse or analyze the input. If they did, the sense that they were actually understanding what they processed would be greatly heightened (whether or not it would still be, as some insist, an illusion). As it is, if a computer says, "I see what you mean" in response to input, there is a strong temptation to dismiss the assertion as an obvious fraud."

People are able to combine the processing of language and vision with apparent ease. In particular, people can use words to describe a picture, and can reproduce a picture from a language description. Moreover, people can exhibit this kind of behaviour over a very wide range of input pictures and language descriptions. Even more impressive is the fact that people can look at images and describe not just the image itself but a set of abstract emotions evoked by it. Although there are theories of how we process vision and language, there are few theories about how such processing is integrated. There have been large debates in Psychology and Philosophy with respect to the degree to which people store knowledge as propositions or pictures (Kosslyn and Pomerantz 1977, Pylyshyn 1973).

There are at least two advantages of linking the processing of natural languages to the processing of visual scenes. First, investigations into the nature of human cognition may benefit. Such investigations are being conducted in the fields of Psychology, Cognitive Science, and Philosophy. Computer implementations of integrated VP and NLP can shed light on how people do it. Second, there are advantages for real-world applications. The combination of two powerful technologies promises new applications: automatic production of speech/text from images; automatic production of images from speech/text; and the automatic interpretation of images with speech/text. The theoretical and practical advantages of linking natural language and vision processing have also been described in Wahlster (1988).

Early work for synthesising simple text from images was conducted by Waltz (1975) who produced an algorithm capable of labelling edges and corners in images of polyhedra. The labelling scheme obeys a constraint minimisation criterion so that only sets of consistent labellings are used. The system can be expected to become 'confused' when presented with an image where two mutually exclusive but self-consistent labellings are possible. This is important because in this respect the program can be regarded as perceiving an illusion such as what humans see in the Necker cube. However, the system seemed to be incapable of any higher-order text descriptions. For example, it did not produce natural language statements such as "There is a cube in the picture."

A number of natural language systems for the description of image sequences have been developed (Herzog and Retz-Schmidt 1990, Neumann and Novak 1986). These systems can verbalize the behaviour of human agents in image sequences about football and describe the spatio-temporal properties of the behaviour observed. Retz-Schmidt (1991) and Retz-Schmidt and Tetzlaff (1991) describe an approach which yields plan hypotheses about intentional entities from spatio-temporal information about agents. The results can be verbalized in natural language. The system called REPLAI-II takes observations from image sequences as input. Moving objects from two-dimensional image sequences have been extracted by a vision system (Herzog et al. 1989) and spatio-temporal entities (spatial

relations and events) have been recognised by an event-recognition system. A focussing process selects interesting agents to be concentrated on during a plan-recognition process. Plan recognition provides a basis for intention recognition and plan-failure analysis. Each recognised intentional entity is described in natural language. A system called SOCCER (André et al. 1988, Herzog et al. 1989) verbalizes real-world image sequences of soccer games in natural language and REPLAI-II extends the range of capabilities of SOCCER. Here, NLP is used more for annotation through text generation with less focus on analysis.

Maaβ et al. (1993) describe a system, called *Vitra Guide*, that generates multimodal route descriptions for computer assisted vehicle navigation. Information is presented in natural language, maps and perspective views. Three classes of spatial relations are described for natural language references: (1) topological relations (e.g. in, near), (2) directional relations (e.g. left, right) and (3) path relations (e.g. along, past). The output for all presentation modes relies on one common 3D model of the domain. Again, Vitra emphasizes annotation through generation of text, rather than analysis, and the vision module considers interrogation of a database of digitized road and city maps rather than vision analysis.

Some of the engineering work in NLP focusses on the exciting idea of incorporating NLP techniques with speech, touchscreen, video and mouse to provide advanced multimedia interfaces (Maybury 1993, Maybury and Wahlster 1998). Examples of such work are found in the ALFresco system which is a multimedia interface providing information on Italian Frescoes (Carenini et al. 1992 and Stock 1991), the WIP system that provides information on assembling, using, and maintaining physical devices like an expresso machine or a lawnmower (André and Rist 1992 and Wahlster et al. 1993), and a multimedia interface which identifies objects and conveys route plans from a knowledge-based cartographic information system (Maybury 1991).

Others developing general IntelliMedia platforms include *Situated Artificial Communicators* (Rickheit and Wachsmuth 1996), *Communicative Humanoids* (Thórisson 1996, 1997), *AESOPWORLD* (Okada 1996, 1997) and MultiModal Interfaces like *INTERACT* (Waibel et al. 1996) and these are discussed further in Chapter 9. Other recent moves towards integration are reported in Denis and Carfantan (1993), Mc Kevitt (1994, 1995/96) and Pentland (1993).

1.2 IntelliMedia 2000+

The Institute for Electronic Systems at Aalborg University, Denmark has expertise in the area of IntelliMedia and has already established an initiative on Multimodal and Multimedia User Interfaces (MMUI) called IntelliMedia 2000+ by the Faculty of Science and Technology (FaST). IntelliMedia 2000+ coordinates research on the production of a number of real-time demonstrators exhibiting examples of IntelliMedia applications, established a new Master's degree in IntelliMedia, and coordinates a nation-wide MultiMedia Network (MMN) concerned with technology transfer to industry. IntelliMedia 2000+ is coordinated from the Center for PersonKommunikation (CPK) which has a wealth of experience

and expertise in spoken language processing, one of the central components of IntelliMedia, but also radio communications which would be useful for mobile applications (CPK Annual Report (1998)). More details on IntelliMedia 2000+ can be found on WWW: http://www.cpk.auc.dk/CPK/MMUI/.

IntelliMedia 2000+ involves four research groups from three Departments within the Institute for Electronic Systems: Computer Science (CS), Medical Informatics (MI), Laboratory of Image Analysis (LIA) and Center for PersonKommunikation (CPK), focusing on platforms for integration and learning, expert systems and decision taking, image/vision processing, and spoken language processing/sound localisation respectively. The first two groups provide a strong basis for methods of integrating semantics and conducting learning and decision taking while the latter groups focus on the two main input/output components of IntelliMedia, vision and speech/sound.

1.2.1 Computer Science (CS)

Research at CS includes computer systems and the design/implementation of programming languages and environments.

Of particular interest for MultiMedia are the following subjects: principles for hypermedia construction, theories of synchronisation and cognition, distributed systems and networking, high volume databases, and the design and use of language mechanisms based on conceptual modelling. Furthermore, CS has a strong research tradition within the interplay between humans, organisations and information systems, and also within the subject of decision support systems and communicating agents, which is highly relevant for emerging research on models for user/system interaction.

CS contributions include experiments for performance evaluation of the available technology (e.g. high speed networking) and experiments on the methodology for design of MultiMedia systems. These contributions are based on existing research activities, which include networks, distributed learning models (Topsy), and prototype hypermedia environments.

In the long term perspective, CS will contribute models for intelligent human-computer interfaces and fundamental understanding of languages/dialogues, graphic elements, etc. based on conceptual understanding, and with implementations of these models. Such models are indispensable for the construction of efficient MultiMedia systems. Also, contributions will be made on efficient techniques for storing high-volume MultiMedia data. Cases will include remote interactive MultiMedia teaching based on existing remote teaching.

Finally, CS will be able to contribute, with technology they have developed, to synchronize both multiple media streams and their content. It is worth noting that several major actors in Intelligent MultiMedia have identified synchronisation of processes as the central technical problem. CS can supply this technology for IntelliMedia 2000+.

1.2.2 Medical Informatics (MI)

The research in the Medical Decision Support System group is centered around medical knowledge-based systems and the development of general tools such as HUGIN (Jensen (F.) 1996), based on Bayesian Networks (Jensen (F.V.) 1996), to support complex decision making.

The research is building on a theory for representing causal dependencies by graphs (Bayesian networks), and uses these to propagate probability estimates. The group has developed several successful medical decision support systems, including sophisticated human-computer interaction issues. A central part of the theoretical development of this paradigm, seen in a global perspective, has taken place at Aalborg University, mainly within the research programme ODIN (Operation and Decision support through Intensional Networks) (a Danish PIFT (Professionel Informatik i Forskning og Teknologi) framework project).

The knowledge-based system technology based on Bayesian networks allowing for a proper handling of uncertain information has shown itself to be usable in creating intelligent coupling between interface components and the underlying knowledge structure. This technology may be integrated in IntelliMedia systems. The Bayes network paradigm, as developed in Aalborg, is already in practical use in user interfaces such as in *Intelligence*, a user and environment context sensitive help system in the major word processing and spreadsheet products from Microsoft.

It is foreseen that IntelliMedia systems will play a central role in the dissemination of information technology in the medical informatics sector. Systems representing complex knowledge, models and data structures e.g. advanced medical diagnostics system, virtual operation room, the telemedical praxis and so on, will require use of knowledge-based techniques for efficient interfacing.

1.2.3 Laboratory of Image Analysis (LIA)

The research at LIA is directed towards three areas: Systems for computer vision, computer vision for autonomous robots, and medical and industrial application of image analysis.

Research within all three areas is sponsored by national and international (EU ESPRIT) research programmes. The main emphasis has been development of methods for continual interpretation of dynamically changing scenes. Example applications include surveillance of in-door and out-door scenes, vision-guided navigation, and interpretation of human and machine manipulation.

Research projects concern extraction of features for description of actions in an environment (i.e. the movement of people, fish, and blood cells) and utilising these descriptions for recognition, monitoring and control of actuators such as mobile robots (safe movements in a dynamically changing environment). This includes recognising and tracking dynamically changing objects, such as hands and human bodies, which has applications in IntelliMedia systems.

So far the research has referred to sensory processing using single modalities, but it

seems obvious that the available methods may be integrated into multi-modal systems, where a major objective is coordination and optimal use of available modalities. New IntelliMedia systems may also include much more flexible modes of interaction between computers, including both speech, body movements, gestures, facial expressions and sign language. This motivates/reinforces the research in interpretation of manipulation and description of dynamically changing objects. Issues of research include also use of the combination of live images and computer graphics for creation of enhanced reality systems, which for example may be used in medical informatics systems on for example medical MRI (Magnetic Resonance Images) images of brain tissue, tele presence systems, and tele manipulation.

1.2.4 Center for PersonKommunikation (CPK)

Speech is the most natural means of communication between humans and between humans and computers. Speech (and other input modalities) has to be processed up to the semantic level in order to enable the computer to understand the intended interaction of the human user - i.e. the computer system must intelligently resolve possible ambiguities and application oriented questions by being able to perform grammatical analyses and consult a database which is able to support decisions – and finally future systems will undoubtedly be requested for availability anywhere and at any time. Continuous speech recognition system application demonstrators that can recognise, understand and react upon specific and limited vocabularies have been available for some time now (Bækgaard 1996, Bækgaard et al. 1992, 1995, Fraser and Dalsgaard 1996, Larsen 1996).

CPK is a research centre which is financially supported by the Danish Technical Research Council and the Faculty of Science and Technology at Aalborg University (see WWW: http://www.cpk.auc.dk/CPK and CPK Annual Report 1998). Research at CPK is focused within the following three areas: Spoken Language Dialogue Systems, Data Communications and Radio Communications. The research within Spoken Language Dialogue Systems has for a long time been focused on human-computer interfacing and interaction and to a large extent been developed in connection with ESPRIT and nationally funded projects. The results obtained so far are of high relevance to many foreseen practical MultiMedia applications and to EU Framework V, and they may advantageously be utilised as partial basis for all activities of IntelliMedia 2000+. The research so far has been focused on the engineering design and development of IntelliMedia for speech and language in the context of professional use. The research is now ready to be further extended into the subsequent research paradigm which is based on the use of a number of available user interface components such as pen-based character recognition, optical character recognition, bar code readers, speech recognition, images and text and by combining these into an integrated MultiMedia interface (e.g. report generation, Personal Data Assistants (PDAs)).

The functionality of CPK research and software platforms is designed with focus on specification of new MultiMedia/MultiModal applications not necessarily by engineering experts but rather those having a professional background in an application domain. So, for example a travel agent would be able to use our platforms to build a spoken dialogue

system for answering questions about flight information without needing to know the ins and outs of the platform itself. To that end the CPK already has experience developing a Dialogue Specification, Design and Management tool called Generic Dialogue System (GDS) (Bækgaard 1996, Dalsgaard and Bækgaard 1994) which has been tested in a number of applications. GDS is now being redesigned into a second generation platform called Dialogue Creation Environment (DCE) (Bai et al. 1998) with an architecture as shown in Figure 1.1. Both GDS and DCE are platforms have been used for applications concerned with spoken input only and a goal is now to see how DCE can be used or extended in a multimodal framework.

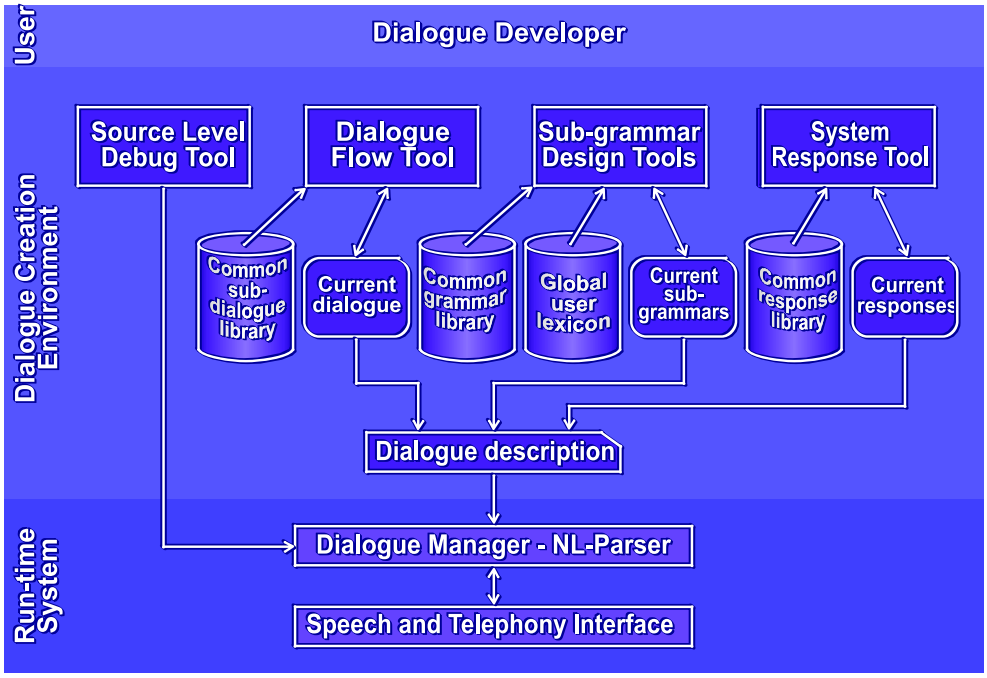

Figure 1.1: Dialogue Creation Environment (DCE)

Two potential future application scenarios are envisaged: (1) for use in offices, an integrated IntelliMedia interface will be investigated; (2) aspects of this IntelliMedia interface will be considered for incorporation in a handheld computer system (e.g. the Personal Data Assistant) permitting direct data capture, as a computer version of a notebook. A major goal of the research is to transfer aspects of the IntelliMedia interface to mobile handheld computer systems.

A basic position taken in this research is that the separate interfacing technologies have already reached a stage of development where it will be possible to use them, with specific and identifiable extension of capabilities, to create an integrated IntelliMedia user interface featuring a spoken human-computer dialogue. It is expected that such dialogue engineering research will form the basis for many future computer systems.

1.3 Education

Teaching is a large part of IntelliMedia 2000+ and two new courses have been initiated: (1) *MultiModal Human Computer Interaction*, and (2) *Readings in Advanced Intelligent MultiMedia*. MultiModal HCI, including traditional HCI, involves teaching of methods for the development of optimal interfaces through methods for layout of buttons, menus, and form filling methods for screens but also includes advanced interfaces using spoken dialogue and gesture. The course on Readings in Advanced Intelligent MultiMedia is innovative and new and includes active learning where student groups present state of the art research papers and invited guest lecturers present their research from IntelliMedia 2000+. A new Master's Degree (M.Eng./M.Sc.) has been established and incorporates the courses just mentioned as core modules of a 1 and 1/2 year course taught in English on IntelliMedia. More details can be found on WWW: http://www.kom.auc.dk/ESN/masters. Occasionally, a Lifelong Learning course is given for returning students of Aalborg University who wish to continue their education. This course is a compression of the core IntelliMedia courses.

The emphasis on group organised and project oriented education at Aalborg University (Kjærsdam and Enemark 1994) is an excellent framework in which IntelliMedia, an inherently interdisciplinary subject, can be taught. Groups can even design and implement a smaller part of a system which has been agreed upon between a number of groups. It is intended that there be a tight link between the education and research aspects of IntelliMedia 2000+ and that students can avail of software demonstrators and platforms developed but can also become involved in developing them. A number of student projects related to IntelliMedia 2000+ have already been completed (Bakman et al. 1997a, 1997b, Nielsen 1997, Tuns and Nielsen 1997) and currently five student groups are enrolled in the Master's conducting projects on multimodal interfaces, billard game trainer, virtual steering wheel, audio-visual speech recognition, and face recognition.

1.4 Choosing a demonstrator

The results from the research groups of IntelliMedia 2000+ have hitherto to a large extent been developed within the groups themselves. However, our goal was to establish collaboration among the groups in order to integrate their results into developing IntelliMedia demonstrator systems and applications. Some of the results would be integrated within a short term perspective as some of the technologically based modules are already available, others on the longer term as new results become available. We set about deciding what our research demonstrator would be by formulating its requirements.

1.4.1 Requirements

We formulated the following goals for the demonstrator:

(1) to demonstrate the actual integration, however primitive, of speech and image modalities. This is the key goal of the project as a whole, and therefore, though of a much more limited/indicative character, of the demonstrator.

(2) to demonstrate the social/political ability to combine our efforts across political and disciplinary boundaries.

(3) to highlight necessary and/or interesting research issues and directions, i.e. the demonstrator is not an end in itself.

(4) to produce useful/working technology - software & hardware - for the subsequent phases of the project, both for research and education where the technology would be general enough to be useful for a number of applications. The resultant platform would be available for postgraduate student projects.

(5) to produce a working concrete demonstrator.

We also considered that the demonstrator application should maximise the following criteria: exploitation of local expertise, commercial potential, incremental achievement, internal collaboration, external collaboration, psychological appeal, and technological relevance. We decided it would not be a requirement that the demonstrator be "interesting" in its own right, or that it necessarily itself be the object of further development. On the other hand, it should touch as many of the topics/issues of interest to the participants as possible, and be extendible in the direction of greater generality or functionality should this be of interest.

The demonstrator would be a single platform called CHAMELEON with a general architecture of communicating agent modules processing inputs and outputs from different modalities and each of which could be tailored to a number of application domains. CHAMELEON would demonstrate that existing platforms for distributed processing, decision taking, image processing, and spoken dialogue processing could be interfaced to the single platform and act as communicating agent modules within it. CHAMELEON would be independent of any particular application domain.

1.4.2 Candidate applications

In general, applications within IntelliMedia may conceptually be divided into a number of broad categories such as intelligent assistant applications, teaching, information browsers, database-access, command control and surveillance, and transaction services (banking). Examples of applications which may result within a short term perspective are enhanced reality (e.g. library guide), newspaper reader for blind/near-blind people, intelligent manuals, dedicated personal communicator (DPC), diagnosis systems (e.g. medical data processing) and mixed reality (e.g. surgery support systems).

Our next step was to choose an application for CHAMELEON. A number of candidate applications were selected and discussed during the course of a number of meetings. These are listed below:

Processing sign language

A system for interpreting and/or generating sign language with speech input/output; would involve vision processing of hand movements and even body movements; in the general case would need a representation of concepts and meaning.

Apparatus repair

A system for diagnosis of problems with apparatuses such as circuit boards and generation of spoken descriptions of the status quo; would involve spoken dialogue interaction on the state of the apparatus and its repair and a headmounted vision system to analyse the scene from the user's point of view; would also involve an enhanced reality system for presenting the visual data.

Neuroanatomy system

A system for presentation of the neuroanatomy and diagnosis of diseases/problems; would involve 3D viewing of 3D body parts and applications in training of doctors/nurses and diagnosis. A model and expert system already exists; could involve spoken dialogue interaction; also, could involve processing of spoken medical reports.

Bridge project

A program which plays the game of bridge with its user(s); would involve graphical display and representation of planning strategies and modes of play; could involve spoken dialogue interaction on the state of play and moves.

Angiogram interpretation

A vision system which can process X-ray angiogram pictures (from different planes) and build a 3D reconstruction of the vasculature; would also involve processing of short medical reports on the pictures from either text or speech; could involve spoken dialogue interaction for training/diagnosis/simulation of treatment; point of angiograms is to locate lesions and medical report aids specification of location of lesions.

CAD program

A CAD (computer aided design) program which builds pictures of described scenes for design or interior decoration; would involve moving of objects like chairs and tables and testing to see how they look in different locations; would involve spoken interaction possibly in conjunction with interpretation of simple gestures and pointing.

Remote presence

Over the (local) internet, in real time, relay a visual simulacrum of a remotely observed (or simulated) scene, accompanied by a synthetically realised person/face that describes what is going on via speech (and perhaps sign language); the user should also be able to address the synthetic person by speech and/or pointing (in the simulacrum scene) to change the point of view and/or cause changes in the "real" scene.

IntelliMedia VideoConferencing

Here VideoConferencing would be conducted in an environment where cameras are controlled through the use of gestures and spoken language; examples of utterances would include, "Point to Mike now" or "Point to the man in the red shirt"; the system could also be used in a teaching environment where it would focus on items on an OHP (over head projector) or black/white board.

These applications boiled down to the fact that the different agent modules within CHAMELEON could be applied initially in at least four areas: (1) spoken dialogue on visual scenes (e.g. apparatuses), (2) spoken medical reports (e.g. angiograms, neuroanatomy), (3) images from visual scenes (e.g. apparatuses) and (4) decision taking (e.g. neuroanatomy). The incorporation of the latter into a demonstrator would be new and innovative where most groups involved in IntelliMedia have not achieved or considered this.

We initially decided that the application would be IntelliMedia VideoConferencing but thought that the vision component would prove too difficult and also the spoken dialogue would be limited and difficult to separate from other dialogue in the application[1]. We had a rethink and finally decided on the following application which includes many concepts from the others.

IntelliMedia WorkBench

An IntelliMedia WorkBench where things (e.g. 2D/3D models, pictures, objects, gadgets, people, or whatever) are placed on a physical table and the user can interact through the use of spoken dialogue and gestures. The system would respond with spoken dialogue and use a laser as a pointing device. An initial domain would be a Campus Information System providing information on 2D building plans.

[1] A simplified version of IntelliMedia VideoConferencing application has since been investigated in Bakman et al. (1997a).

Chapter 2

CHAMELEON and the IntelliMedia WorkBench

The four groups of IntelliMedia 2000+ have developed the first prototype of an IntelliMedia software and hardware platform called CHAMELEON which is general enough to be used for a number of different applications. CHAMELEON demonstrates that existing software modules for (1) distributed processing and learning, (2) decision taking, (3) image processing, and (4) spoken dialogue processing can be interfaced to a single platform and act as communicating agent modules within it. CHAMELEON is independent of any particular application domain and the various modules can be distributed over different machines. Most of the modules are programmed in C++ and C.

CHAMELEON demonstrates that (1) it is possible for agent modules to receive inputs particularly in the form of images and spoken dialogue and respond with required outputs, (2) individual agent modules can produce output in the form of semantic representations, (3) the semantic representations can be used for effective communication of information between different modules, and (4) various means of synchronising the communication between modules can be tested to produce optimal results.

2.1 IntelliMedia WorkBench

An initial application of CHAMELEON is the *IntelliMedia WorkBench* which is a hardware and software platform as shown in Figure 2.1. One or more cameras and lasers are mounted in the ceiling, a microphone array placed on the wall and there is a table where things (2D/3D models, building plans, pictures, objects, gadgets, people, or whatever) can be placed.

The current domain is a *Campus Information System* which at present gives information on the architectural and functional layout of a building. 2D architectural plans of the building drawn on white paper are laid on the table and the user can ask questions about them. At present the plans represent two floors of the 'A' (A2) building at Fredrik Bajers Vej 7, Aalborg University. Extensions of this application could include a 3D model and

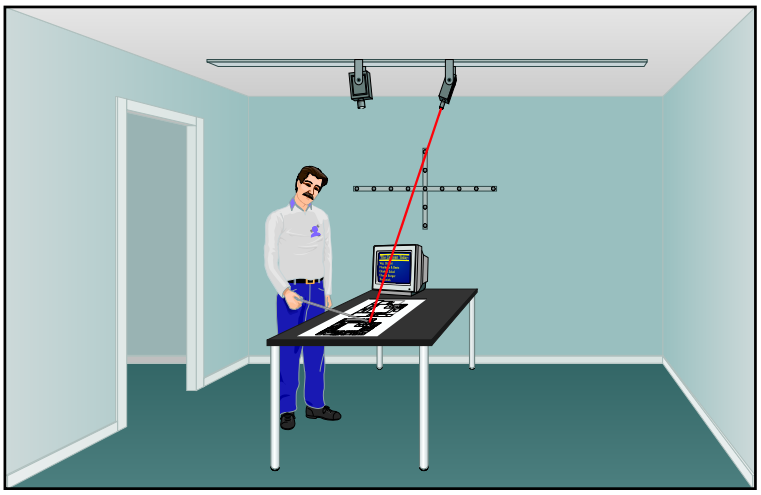

Figure 2.1: Physical layout of the IntelliMedia WorkBench

other similar domains would be hospitals or town halls, or a model of a city for tourist information. Moving up to a 3D model would involve at least two cameras and more complex 3D vision processing.

Presently, there is one static camera which calibrates the plans on the table and the laser, and interprets the user's pointing while the system points to locations and draws routes with a laser. Inputs are simultaneous speech and/or pointing gestures and outputs are synchronised speech synthesis and pointing. We currently run all of CHAMELEON on a 200 MHz Intel pentium computer under Linux which handles input for the Campus Information System in real-time. It displays information about CHAMELEON's processing and could also display data relevant to the domain such as internet/WWW pages for people or locations referred to.

More specifically, the WorkBench includes three tables placed close together and giving a working area of 185x120cm (see Figure 2.2). The tables are covered with a black cloth. The static camera is mounted in the ceiling 260cm above the physical table and looking down at it. Details on the camera can be found in Appendix A. The laser is mounted next to the camera as can be seen in Figure 2.3.

The 2D plan, which is placed on the table, is printed out on A0 paper having the dimensions: 84x118cm. Due to the size of the pointer's tip (2x1cm), the size of the table, the resolution of the camera and uncertainty in the tracking algorithm, a size limitation is introduced. The smallest room in the 2D plan, which is a standard office, can not be less than 3cm wide. The size of a standard office on the printout is 3x4cm which is a feasible size for the system. The 2D plan is shown in Figure 2.4.

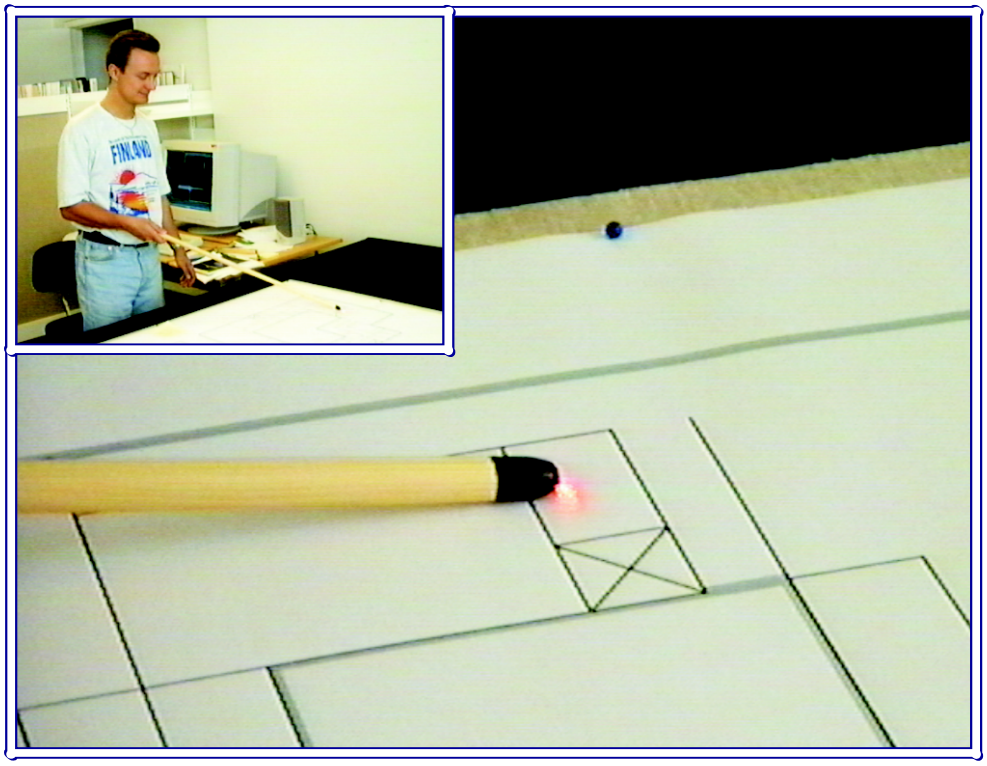

Figure 2.2: Physical table

2.2 Sample dialogue interaction

We present here a sample dialogue which the current first prototype can process. The example includes user intentions which are instructions and queries, and exophoric/deictic reference.

```
USER:       Show me Tom's office.
CHAMELEON:  [points]
            This is Tom's office.
USER:       Point to Thomas' office.
CHAMELEON:  [points]
            This is Thomas' office.
USER:       Where is the computer room?
CHAMELEON:  [points]
            The computer room is here.
USER:       [points to instrument repair]
            Whose office is this?
CHAMELEON:  [points]
            This is not an office, this is instrument repair.
USER:       [points]
            Whose office is this?
```

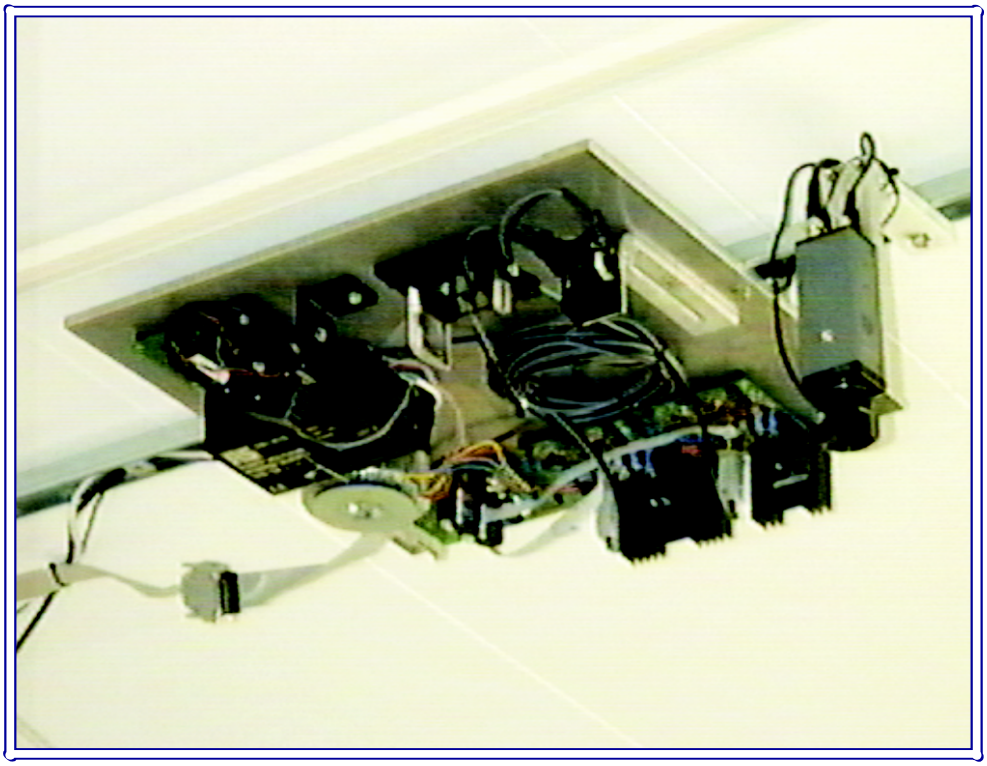

Figure 2.3: Laser and camera

```
CHAMELEON: [points]
           This is Paul's office.
USER:      Show me the route from Lars Bo Larsen's office to Hanne Gade's office.
CHAMELEON: [draws route]
           This is the route from Lars Bo's office to Hanne's office.
USER:      Show me the route from Paul Mc Kevitt's office
           to instrument repair.
CHAMELEON: [draws route]
           This is the route from Paul's office to instrument repair.
USER:      Show me Paul's office.
CHAMELEON: [points]
           This is Paul's office.
```

CHAMELEON can process deictic reference ("Whose office is *this*?") which is one of the most frequently occurring phenomena in IntelliMedia. However, spatial relations are another phenomenon occurring regularly which we do not yet address (e.g. "Who's in the office *beside* him?"). Also, note that CHAMELEON assumes Paul Dalsgaard as default Paul[1] although there are two Pauls. A later prototype of the system should become active here and ask the user a question by first pointing out that there are two Pauls and

[1] This is because Paul Dalsgaard is more senior :).

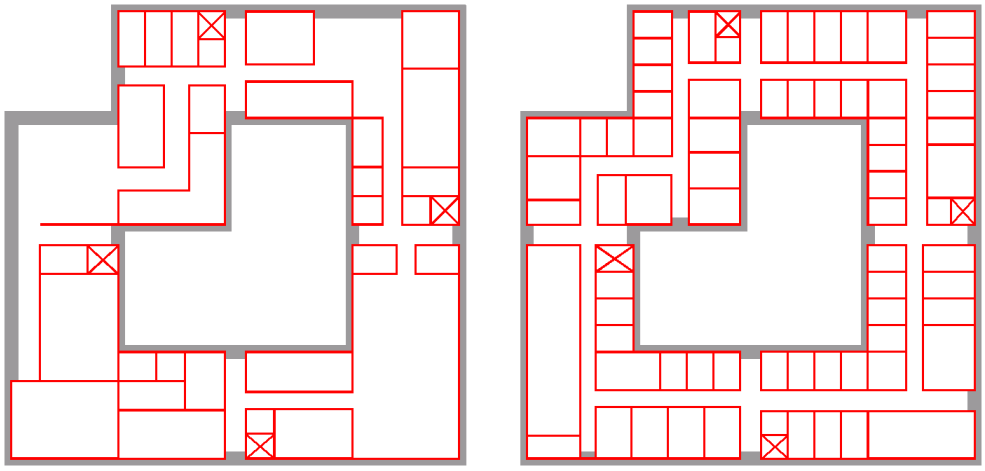

Figure 2.4: 2D plan of the 'A' building at Fredrik Bajers Vej 7, Aalborg University. Left: ground floor; Right: 1st floor.

then asking which does he/she mean. Here is a sample dialogue including the additional phenomena of spatial relations (beside), ambiguity resolution, statements/declaratives, ellipsis, and domain model update, planned for the next version of the system.

```
USER:      Show me Paul's office.
CHAMELEON: [points(twice)]
           This is Paul Dalsgaard's office and this is Paul Mc Kevitt's office.
or/
CHAMELEON: There are two Paul's. Do you mean Paul Dalsgaard or Paul Mc Kevitt?
USER:      Paul Dalsgaard!
CHAMELEON: [points]
           This is Paul Dalsgaard's office.

USER:      Who's in the office beside him?
CHAMELEON: [points]
           Boerge, Jorgen and Hanne's offices are beside Paul Dalsgaard's office.
USER:      [points]
           Whose office is this?
CHAMELEON: [points]
           This is Ipke's office.
USER:      No, that's Tom's office!
CHAMELEON: [points]
           I've updated Ipke's office to Tom's office.
/or
CHAMELEON: [points]
           Ipke and Tom are in the same office!
```

Note that in this example a record of the meaning representations in the dialogue history

becomes important because it is then used to resolve referents like "Paul Dalsgaard" for "him". More complex processing would involve recording of user intentions over time and noticing that when a user repeats an intention this indicates dissatisfaction with a response. Also, monitoring of previous dialogue context aids resolution of ambiguity in user input. For example, it may be unclear whether a user is pointing to one as opposed to another while asking "Whose office is this?" but the fact that the user has already asked about one of them in the previous dialogue probably indicates that the current focus is the other one. Of course the system can again become active in the dialogue and ask the user for a clarification as to which office he/she means. Also, there are other spatial relations such as "left", "right", "up", "down", "beside" and queries like "Who's in the office two up from him?"

2.3 Architecture of CHAMELEON

CHAMELEON has a distributed architecture of communicating agent modules processing inputs and outputs from different modalities and each of which can be tailored to a number of application domains. It is being developed in both a top-down and bottom-up manner making sure it is general enough for multiple application domains but at the same time keeping particular domains in mind. An open architecture has been chosen to allow for easy integration of new modules. The process synchronisation and intercommunication for CHAMELEON modules is performed using the DACS (Distributed Applications Communication System) Inter Process Communication (IPC) software (see Fink et al. 1995, 1996) which enables CHAMELEON modules to be glued together and distributed across a number of servers.

Presently, there are ten software modules in CHAMELEON: blackboard, dialogue manager, domain model, gesture recogniser, laser system, microphone array, speech recogniser, speech synthesiser, natural language processor (NLP), and Topsy as shown in Figure 2.5. The blackboard and dialogue manager form the kernel of CHAMELEON. The modules are a mixture of commercially available products (e.g. speech recogniser and synthesiser), custom made products (e.g. laser system) and modules developed by the IntelliMedia 2000+ project team (e.g. gesture recogniser and NLP module). Some modules such as the laser pointer and microphone array are simply interfaces for plug in hardware modules whereas other modules are more involved with processing semantic representations. Of course, the data in the domain model needs to be changed for different domains. The microphone array is a functioning module and is in the process of being integrated with CHAMELEON. We shall now give a brief description of each module.

Blackboard

The blackboard stores semantic representations produced by each of the other modules and keeps a history of these over the course of an interaction. All modules communicate through the exchange of semantic representations with each other or the blackboard. Semantic

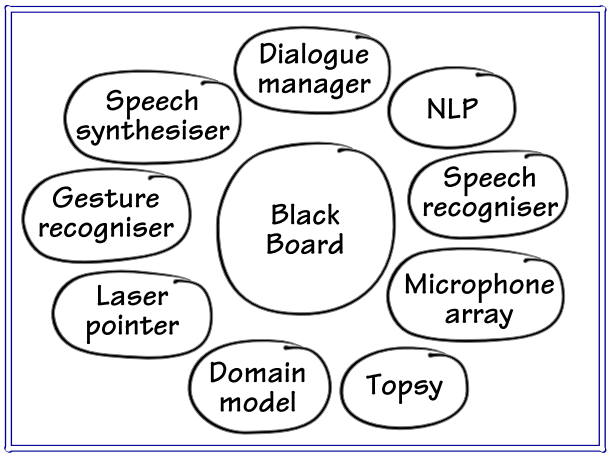

Figure 2.5: Architecture of CHAMELEON

representations are frames in the spirit of Minsky (1975) and our frame semantics consists of (1) input, (2) output, and (3) integration frames for representing the meaning of intended user input and system output. The intention is that all modules in the system will produce and read frames. Frames are coded in CHAMELEON as messages built of predicate-argument structures following the BNF definition given in Appendix D.

Dialogue manager

The dialogue manager makes decisions about which actions to take and accordingly sends commands to the output modules (laser and speech synthesiser) via the blackboard. At present the functionality of the dialogue manager is to integrate and react to information coming in from the speech/NLP and gesture modules and to sending synchronised commands to the laser system and the speech synthesiser modules. Phenomena such as managing clarification subdialogues where CHAMELEON has to ask questions are not included at present. It is hoped that in future prototypes the dialogue manager will enact more complex decision taking over semantic representations from the blackboard using, for example, the HUGIN software tool (Jensen (F.) 1996) based on Bayesian Networks (Jensen (F.V.) 1996).

Domain model

The domain model contains a database of all locations and their functionality, tenants and coordinates. The model is organised in a hierarchical structure: areas, buildings and rooms. Rooms are described by an identifier for the room (room number) and the type of the room (office, corridor, toilet, etc.). For offices there is also a description of tenants by a number of attributes (first and second name, title, affiliation, etc.). The model includes functions

that return information about a room or a person. Possible inputs are coordinates or room number for rooms and name for persons, but in principle any attribute can be used as key and any other attribute can be returned. Furthermore, a path planner is provided, calculating the shortest route between two locations.

Gesture recogniser

A design principle of imposing as few physical constraints as possible on the user (e.g. data gloves or touch screens) leads to the inclusion of a vision based gesture recogniser. Currently, it tracks a pointer via a camera mounted in the ceiling. Using one camera, the gesture recogniser is able to track 2D pointing gestures in real time. Only two gestures are recognised at present: pointing and not-pointing. The recognition of other more complex kinds of gestures like marking an area and indicating a direction (with hands and fingers) will be incorporated in the next prototype.

The camera continuously captures images which are digitised by a frame-grabber. From each digitised image the background is subtracted leaving only the motion (and some noise) within this image. This motion is analysed in order to find the direction of the pointing device and its tip. By temporal segmenting of these two parameters, a clear indication of the position the user is pointing to at a given time is found. The error of the tracker is less than one pixel (through an interpolation process) for the pointer.

Laser system

A laser system acts as a "system pointer". It can be used for pointing to positions, drawing lines and displaying text. The laser beam is controlled in real-time (30 kHz). It can scan frames containing up to 600 points with a refresh rate of 50 Hz thus drawing very steady images on surfaces. It is controlled by a standard Pentium PC host computer. The pointer tracker and the laser pointer have been carefully calibrated so that they can work together. An automatic calibration procedure has been set up involving both the camera and laser where they are tested by asking the laser to follow the pointer.

Microphone array

A microphone array (Leth-Espensen and Lindberg 1995, 1996) shown in Figure 2.6 is used to locate sound sources, e.g. a person speaking. Depending upon the placement of a maximum of 12 microphones it calculates sound source positions in 2D or 3D. It is based on measurement of the delays with which a sound wave arrives at the different microphones. From this information the location of the sound source can be identified. Another application of the array is to use it to focus at a specific location thus enhancing any acoustic activity at that location. This module is in the process of being incorporated into CHAMELEON.

Figure 2.6: Microphone array experimental setup

Speech recogniser

Speech recognition is handled by the grapHvite real-time continuous speech recogniser (Power et al. 1997). It is based on HMMs (Hidden Markov Models) of triphones for acoustic decoding of English or Danish. The recognition process focusses on recognition of speech concepts and ignores non content words or phrases. A finite state network describing phrases is created by hand in accordance with the domain model and the grammar for the natural language parser. The latter can also be done automatically by a grammar converter in the NLP module. The speech recogniser takes speech signals as input and produces text strings as output. Integration of the latest CPK speech recogniser (see Christensen et al. 1998a,b) which is under development is being considered.

Speech synthesiser

We use the Infovox Text-To-Speech (TTS) speech synthesiser which at present is capable of synthesising Danish and English (Infovox 1994). It is a rule based formant synthesiser and can simultaneously cope with multiple languages, e.g. pronounce a Danish name within an English utterance. Infovox takes text as input and produces speech as output. Integration of the CPK speech synthesiser (Jensen et al. 1998, Nielsen et al. 1997) which is under development for English is being considered.

Natural language processor (NLP)

The natural language parser is based on a compound feature based (so-called unification) grammar formalism for extracting semantics from the one-best utterance text output from the speech recogniser (Brøndsted 1998). The parser carries out a syntactic constituent analysis of input and subsequently maps values into semantic frames. The rules used for syntactic parsing are based on a subset of the EUROTRA formalism, i.e. in terms of lexical rules and structure building rules (Bech 1991). Semantic rules define certain syntactic subtrees and which frames to create if the subtrees are found in the syntactic parse trees. For each syntactic parse tree the parser generates only one predicate and all semantic frames created are arguments or sub-arguments of this predicate. If syntactic parsing cannot complete, the parser can return the found frame fragments to the blackboard.

The natural language generator is currently under construction and at present generation is conducted by using canned text. The generator will use the same grammar definitions as the parser and can in terms of input-output be considered the reverse counterpart of the parser.

Topsy learner

The basis of the Phase Web paradigm (see Manthey 1998a,b), and its incarnation in the form of a program called Topsy, is to represent knowledge and behaviour in the form of hierarchical relationships between the mutual exclusion and co-occurrence of events. In AI parlance, Topsy is a distributed, associative, continuous-action, dynamic partial-order planner that learns from experience. Relative to MultiMedia, integrating independent data from multiple media begins with noticing that what ties otherwise independent inputs together is the fact that they occur simultaneously (more or less). This is also Topsy's basic operating principle, but this is further combined with the notion of mutual exclusion, and thence to hierarchies of such relationships (see Manthey 1998b).

2.4 DACS

DACS is currently the communications system for CHAMELEON and the IntelliMedia WorkBench and is used to glue all the modules together enabling communication between them. Applications of CHAMELEON typically consist of several interdependent modules, often running on separate machines or even dedicated hardware. This is indeed the case for the IntelliMedia WorkBench application. Such distributed applications have a need to communicate in various ways. Some modules feed others in the sense that all generated output from one is treated further by another. In the Campus Information System all modules report their output to the blackboard where it is stored. Although our intention is currently to direct all communication through the blackboard, we could just as well have chosen to simultaneously transfer output to several modules. For example, utterances collected by the speech recogniser can be sent to the blackboard but also sent simultaneously to the NLP module which may become relevant when efficiency is an important issue.

Another kind of interaction between processes is through remote procedure calls (RPCs), which can be either *synchronous* or *asynchronous*. By synchronous RPCs we understand procedure calls where we want immediate feedback, that is, the caller stops execution and waits for an answer to the call. In the Campus Information System this could be the dialogue manager requesting the last location to which a pointing event occurred. In the asynchronous RPC, we merely submit a request and carry on with any other task. This could be a request to the speech synthesiser to produce an utterance for the user or to the laser to point to some specific location. These kinds of interaction should be available in a uniform way in a heterogeneous environment, without specific concern about what platform the sender and receiver run on.

All these facilities are provided by the Distributed Applications Communication System (DACS) developed at the University of Bielefeld, Germany (see Fink et al. 1995, 1996), where it was designed as part of a larger research project developing an IntelliMedia platform (Rickheit and Wachsmuth 1996) discussed further in Chapter 9. DACS uses a communication demon on each participating machine that runs in user mode, allows multiple users to access the system simultaneously and does not provide a virtual machine dedicated to a single user. The demon acts as a router for all internal traffic and establishes connections to demons on remote machines. Communication is based on simple asynchronous message passing with some extensions to handle dynamic reconfigurations of the system during runtime. DACS also provides on top more advanced communication semantics like RPCs (synchronous and asynchronous) and *demand streams* for handling data parts in continuous data streams. All messages transmitted are recorded in a Network Data Representation which includes type and structure information. Hence, it is possible to inspect messages at any point in the system and to develop generic tools that can handle any kind of data. DACS uses POSIX threads to handle connections independently in parallel. A database in a central name service stores the system configuration to keep the network traffic low during dynamic reconfigurations. A DACS Debugging Tool (DDT) allows inspection of messages before they are delivered, monitoring configurations of the system, and status on connections.

2.5 Summary

The establishing of CHAMELEON and the IntelliMedia WorkBench will promote a variety of student projects as both students working on design of general applications taking advantage of the previously established modules and students focusing on a single, more detailed problem (e.g. design of microphone arrays) can use them as the basis of their projects. Furthermore, work on many different applications and domains can be carried out simultaneously. It is intended that student projects should be exploited and help enhance functionality of the WorkBench and diversity of applications. One way of ensuring this will be to employ student programmers to document their work thus making it available to other student projects and in general. Students have already been employed on developing software for controlling the laser (Moeslund et al. 1998) and on evaluating and testing

DACS (Nielsen 1997).

Chapter 3

Dialogue management

Dialogue management is concerned with managing dialogue between users and the system. The kernel modules of CHAMELEON which constitute the blackboard, dialogue manager and DACS are those most involved with dialogue management. The blackboard is one of the most important modules in CHAMELEON because it is where the semantics of input, output and integrated information is stored. Other modules aid in constructing the semantics on the blackboard. We have plans for an active blackboard module although presently the blackboard is simply a passive file store of information. The dialogue manager makes decisions about which actions to take and accordingly sends commands to the output modules (laser and speech synthesiser) via the blackboard. The DACS communications system manages the interaction with the user at the systems level.

3.1 Frame semantics

The meaning of interactions over the course of the MultiModal dialogue is represented using a frame semantics with frames in the spirit of Minsky (1975). The intention is that all modules in the system can produce and read frames. Frames are coded in CHAMELEON with messages built as predicate-argument structures following the BNF definition given in Appendix D. Frames represent some crucial elements such as *module*, *input/output*, *intention*, *location*, and *timestamp*. Module is simply the name of the module producing the frame (e.g. NLP). Inputs are the input recognised whether spoken (e.g. "Show me Hanne's office") or gestures (e.g. pointing coordinates) and outputs the intended output whether spoken (e.g. "This is Hanne's office.") or gestures (e.g. pointing coordinates). Timestamps can include the times a given module commenced and terminated processing and the time a frame was written on the blackboard. The frame semantics also includes representations for two key phenomena in language/vision integration: reference and spatial relations.

Frames can be grouped into three categories: (1) *input*, (2) *output* and (3) *integration*. Input frames are those which come from modules processing perceptual input, output frames are those produced by modules generating system output and integration frames are integrated meaning representations constructed over the course of a dialogue (i.e. all

other frames). Here, we shall discuss frames with a focus more on frame semantics than on frame syntax and in fact the actual coding of frames as messages within CHAMELEON has a different syntax (see Appendix D).

3.1.1 Input frames

An input frame takes the general form:

[MODULE
 INPUT: input
 INTENTION: intention-type
 TIME: timestamp]

where MODULE is the name of the input module producing the frame, INPUT can be at least UTTERANCE or GESTURE, *input* is the utterance or gesture and *intention-type* includes different types of utterances and gestures. An utterance input frame can at least have intention-type (1) query?, (2) instruction! and (3) declarative. An example of an utterance input frame is:

[SPEECH-RECOGNISER
 UTTERANCE: (Point to Hanne's office)
 INTENTION: instruction!
 TIME: timestamp]

A gesture input frame is where *intention-type* can be at least (1) pointing, (2) mark-area, and (3) indicate-direction. An example of a gesture input frame is:

[GESTURE
 GESTURE: coordinates (3, 2)
 INTENTION: pointing
 TIME: timestamp]

3.1.2 Output frames

An output frame (F-out) takes the general form:

[MODULE
 INTENTION: intention-type
 OUTPUT: output
 TIME: timestamp]

where MODULE is the name of the output module producing the frame, *intention-type* includes different types of utterances and gestures and OUTPUT is at least UTTERANCE or GESTURE. An utterance output frame can at least have intention-type (1) query? (2) instruction!, and (3) declarative. An example utterance output frame is:

[SPEECH-SYNTHESIZER
INTENTION: declarative
UTTERANCE: (This is Hanne's office)
TIME: timestamp]

A gesture output frame can at least have intention-type (1) description (pointing), (2) description (route), (3) description (mark-area), and (4) description (indicate-direction). An example gesture output frame is:

[LASER
INTENTION: description (pointing)
LOCATION: coordinates (5, 2)
TIME: timestamp]

3.1.3 Integration frames

Integration frames are all those other than input/output frames. An example utterance integration frame is:

[NLP
INTENTION: description (pointing)
LOCATION: office (tenant Hanne) (coordinates (5, 2))
UTTERANCE: (This is Hanne's office)
TIME: timestamp]

Things become even more complex with the occurrence of references and spatial relationships:

[MODULE
INTENTION: intention-type
LOCATION: location
LOCATION: location
LOCATION: location
SPACE-RELATION: beside
REFERENT: person
LOCATION: location
TIME: timestamp]

An example of such an integration frame is:

[DOMAIN-MODEL
INTENTION: query? (who)
LOCATION: office (tenant Hanne) (coordinates (5, 2))
LOCATION: office (tenant Jørgen) (coordinates (4, 2))
LOCATION: office (tenant Børge) (coordinates (3, 1))
SPACE-RELATION: beside
REFERENT: (person Paul-Dalsgaard)
LOCATION: office (tenant Paul-Dalsgaard) (coordinates (4, 1))
TIME: timestamp]

It is possible to derive all the frames produced on a blackboard for example input. Complete blackboard histories for the instruction "Point to Hanne's office" and the query "Whose office is this?" + [pointing] (exophoric/deictic reference) are given in Appendix B. The frames given are placed on the blackboard as they are produced and processed. In these histories we choose to have modules interacting in a completely distributed manner with no single coordinator. The actual current implementation of CHAMELEON has a more top-down coordinating dialogue manager.

3.1.4 Coding of frames

Frames in CHAMELEON are coded as messages with a predicate-argument format and are passed between modules by DACS. A BNF definition of messages, which are text strings, is given in Appendix D. For the sample dialogue given in Chapter 2, Section 2.2 CHAMELEON's actual blackboard history is shown in Appendix C. The use of information held by frame messages necessitates access to individual message slots and because the structure of messages are very general we need a flexible internal representation. This structure represents everything as predicates - all atoms are represented as simple predicates without arguments, but in general a predicate can have an arbitrary number of arguments of any type. This is obtained by designing a structure with a predicate name, a pointer to a list of arguments (if any) and another pointer to link multiple arguments. This structure is hidden from users such that all interaction takes place through a collection of access functions. There are functions that return the type of an argument:

```
is_identifier(predicate)

is_integer(predicate)

is_string(predicate)

is_variable(predicate)
```

`is_predicate(predicate)`

a function that returns a named argument,

`get_argument(predicate, label)`

and, of course, a function to retrieve the value (label) of a predicate,

`get_label(predicate)`

Moreover, functions are provided for conversion between textual format and internal representation. The function,

`predicate_to_string(predicate)`

yields a string representation of the predicate, and the function,

`string_to_predicate(message)`

performs the inverse conversion, resulting in a pointer to a predicate. This function parses the string and builds the internal representation using the functions:

`new_predicate(label)`

`add_argument(predicate, label)`

The code for frame messages is compiled separately and included in other modules which then operate in a rule based fashion on the representation. An example of the use of this code is given in the description of the dialogue manager in Section 3.3.

3.2 Blackboard

Information flow and module communication within CHAMELEON are shown in Figures 3.1 and 3.2. Note that Figure 3.1 does not show the blackboard as a part of the communication but rather the abstract flow of information between modules.

Figure 3.2 shows the actual passing of information between the speech recogniser, NLP module, and dialogue manager. As is shown all information exchange between individual modules is carried out using the blackboard as mediator.

As the intention is that no direct interaction between modules need take place the architecture is modularised and open but there are possible performance costs. However, nothing prohibits direct communication between two or more modules if this is found to be more convenient. For example, the speech recogniser and NLP modules can interact directly as the parser needs every recognition result anyway and at present no other module has use for output from the speech recogniser.

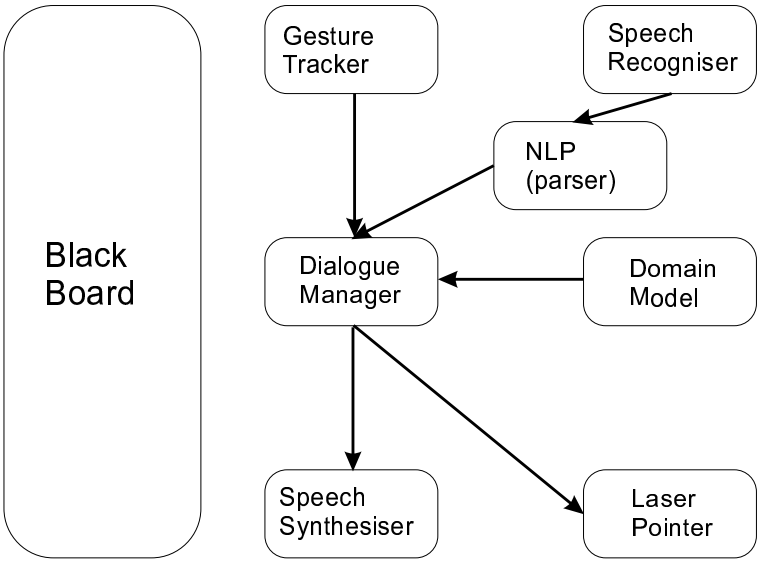

Figure 3.1: Information flow and module communication

3.2.1 Functionality

At present the blackboard is simply a passive file store. However, we have plans for a more active blackboard design as is described here. The blackboard has two tasks:

(1) to act as a repository for all input/output and semantic information in the system

(2) to provide semantic information to other modules when requested

The design and implementation involve data being stored and retrieved using standard database techniques (e.g. SQL). All requests would be in the form of queries to the blackboard. The blackboard accommodates two types of information retrieval:

(1) explicit requests, i.e. when a module submits a query

(2) implicit (or automatic) requests

Explicit requests

When an explicit request is submitted to the blackboard it performs a search and returns the results to the appropriate module. A typical request could be the dialogue manager asking for the tenant(s) at a specific location or a request for the latest known position pointed at.

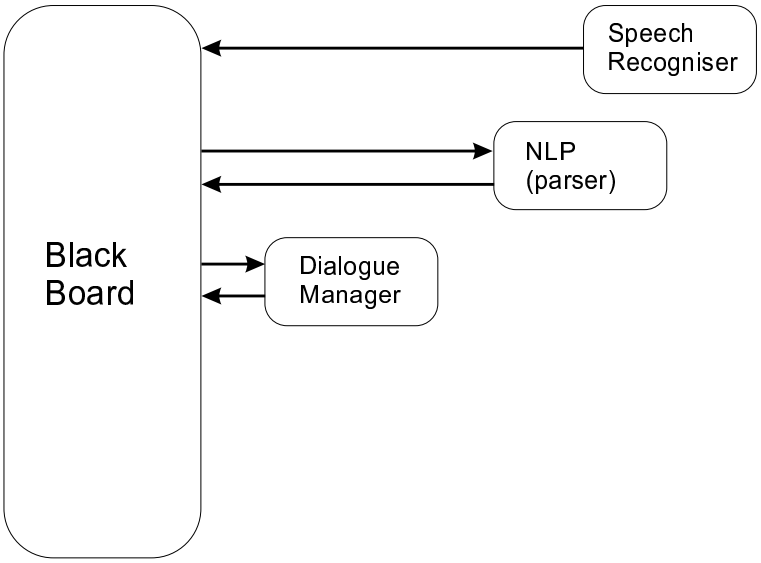

Figure 3.2: Information flow with the blackboard

Implicit requests

Implicit requests are not carried out immediately when submitted to the blackboard but instead they are stored in a table and whenever a new frame of information is received by the blackboard all implicit requests are activated and the search results are returned to the respective submitters. Modules can at any time submit or withdraw implicit requests. A typical implicit request could be the NLP module requesting all frames from the speech recogniser whenever they are sent to the blackboard or the laser module requesting all frames where there is an intention of pointing with coordinates.

Special features

A number of special features could be made available. These include:

Most recent frame: It is possible to request the most recent frame(s) matching a search pattern. Examples of the use of this request would be for conducting anaphoric reference and deictic resolution on the basis of recency. For example, if an utterance with an anaphoric reference occurs, e.g. "Who's in the office beside *him*?" then the dialogue manager or NLP module might submit a request to the blackboard for the most recent frame with mention of a person. Another example is where deictic reference occurs in an utterance, e.g. "Whose office is *this*?" and the dialogue manager or NLP module would submit a request to the blackboard for the latest known position of the pointer. Another example is in a dialogue repair situation when backtracking is needed in order to uncover the point where the error occurred.

Uninstantiated fields: An important concept is open (or uninstantiated) fields. By specifying an open field, by for example using a variable, in a frame a module can express that some information is unknown or sought for. For example, this would be where the NLP module or dialogue manager wants information about the room location for a specific person. They would submit frames as explicit requests containing an uninstantiated location field. At some time, the domain model may have submitted an implicit request specifying that it wants all frames with underspecified locations or names. The underspecified frame from the module concerned will then be sent to the domain model.

Hardwiring: It might be profitable to be able to hardwire common search patterns into the blackboard for greater efficiency and or conveniency. For example, this could be useful in the case of backtracking or for logging and displaying the system state.

3.2.2 Blackboard architecture

A closer look at the blackboard is shown in Figure 3.3 below. It consists of three main parts, namely (1) the database, (2) an interface handling communication to the other system modules and (3) the table of implicit requests. The database is intended to be implemented using mySQL (mySQL 1998) which is a standard SQL database system available for most platforms including Linux and Solaris. The other parts may be implemented in any convenient programming language which can communicate with the DACS ICP software. Our initial thought is to use Java.

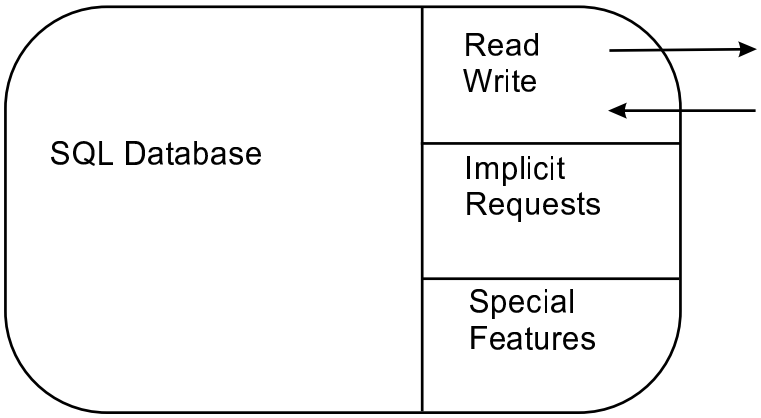

Figure 3.3: Internal blackboard architecture

3.2.3 Status of blackboard

Currently CHAMELEON's blackboard as shown in Appendix C is simply a database of frames stored over the course of a dialogue. Hence, it does not include most of the func-

tionalities discussed above which are currently under implementation. The blackboard acts as a database holding the history of the system in terms of the frames passed between the different modules, and in the current version it also involves an active element directing the messages. This routing is done by hardcoded rules, but as indicated above future versions will probably include a metalanguage enabling individual modules to describe the datastreams and services they provide as well as requests for subscription to datastreams. To get full generality service consumers could then request services in terms of frames, thus relieving the current necessity for knowledge on services at compile time.

3.3 Dialogue manager

The dialogue manager is designed in a modular fashion, where individual sections handle different kinds of interaction. It acts in accordance with the intention of received frames. At the top level it distinguishes between *queries*, *instructions* and *descriptions*. Queries are subdivided into *where* and *who* questions. Frames in each category are analysed with respect to present and absent information, and missing details are then collected from either the domain model (e.g. the type of room at a specific location or the name of a person occupying an office) or the gesture module (the coordinates referred to at a specific time) through RPCs (remote procedure calls). These details are added to frames, and appropriate responses are constructed in terms of frames giving directives to output modules. Instructions are handled in a similar fashion; they are divided into pointing instructions and instructions concerning routes. Likewise, frames are analysed for missing information which is collected and new frames are generated and directed to relevant modules.

The dialogue manager operates on frames coded as messages as described in Section 3.1.4 above. When a frame arrives at the dialogue manager it is converted to an internal representation. For example, in the following a frame from the NLP module is converted,

```
string_to_predicate("nlp(intention(query(who)),
location($v, type(office)),time(8786665070))")
```

and the intention, which is always present, is extracted by,

```
get_argument(predicate, "intention")
```

The value (label) of the intention (e.g. “who”) is fetched by:

```
get_argument(intention, "query")
```

which returns the `"query"` predicate if present and otherwise NULL. Queries are passed on to a query handler, which again determines the subclass “where” or “who” questions, in an analogue way. From the subpredicate location an external reference (`$v`) to an office is found and the absent information is requested from the gesture module through an RPC.

```
gm_get_coordinates(get_label(time))
```

where `time` is the time extracted from the incoming frame. The coordinates are now added to the frame and the type of the room is retrieved from the domain model through the call,

```
dm_get_room_type_from_coordinates(xy)
```

and checked against the specified room type. On success the name of the tenant is retrieved by,

```
dm_get_name_from_coordinates(xy)
```

and an appropriate response is constructed in terms of a frame giving directives to the speech synthesiser and laser output modules. The resultant frame,

```
dialog_manager(output(laser(point(coordinates(696,570))),
speech_synthesizer(utterance("This is not an office,
this is instrument repair"))))
```

is then passed on to the blackboard module for further processing.

The dialogue manager is quite simple at present but in future prototypes of CHAMELEON more sophisticated processing is envisaged where, for example, the Bayesian Network (Jensen (F.V.) 1996) HUGIN software tool (Jensen (F.) 1996) can introduce more complex decision taking over semantic representations.

3.4 DACS and CHAMELEON

DACS is currently the communications system for CHAMELEON and the IntelliMedia WorkBench. In order to secure smooth and consistent use of DACS we have designed a special module to cater for all interaction with it. This is partly to relieve individual module designers from coping with details of DACS and partly due to the fact that our use of DACS only takes advantage of a subset of the provided functionality which allows simplification of the actual interaction. To use the facilities of DACS, all modules have to register, as they have to unregister when work is completed. The registration involves an internal variable and results in a descriptor that has to be used in subsequent calls to DACS. These details are hidden in our own registration procedure, resulting in a simple function call from each module:

```
register_module(<module-name>)
```

where `<module-name>` is a string. This function holds information about the module in an internal structure, and handles deregistration automatically when the calling process stops execution. Likewise the provider of a stream simply registers with the name of a stream:

```
register_stream(<stream-name>)
```

These functions take care of error checks and return 0 if a failure occurs and 1 otherwise. After streams have been established messages are put to them, and as messages are always communicated in textual form the provider simply calls a function `put_to_stream`:

```
put_to_stream(<stream-name>, <message>)
```

The consumer of a stream has to know its name and orders the next message on the stream by,

```
order_stream(<stream-name>)
```

which returns true if there is anything available on the stream. The consumer can then fetch a message by,

```
get_from_stream(<stream-name>, <message>)
```

which results in the actual message being left in the second argument.

The remote procedure calls work in the same fashion. Each provider of a function registers the function and a consumer calls the function together with pointers to arguments and information about formats in order to enable correct encoding and decoding. Again details are hidden from the module designers, that just call simple functions like:

```
speech_synthesizer(<utterance>)
```

DACS has been analysed and ported to Aalborg University where it currently runs on SUN/Unix and PC/Linux platforms as described in Nielsen (1997). DACS seems to be an appropriate tool for our purposes and we expect to exploit its capabilities further in the future. We are also evaluating other communications systems.

3.5 Summary

Dialogue management aspects of CHAMELEON have been in focus here. The blackboard, dialogue manager and DACS comprise CHAMELEON's kernel which is most involved with dialogue management. DACS, which glues all modules together and manages the interaction with the user at a technical level is an important part of the kernel. We have described the blackboard frame semantics which is used to represent the meaning of interactions over the course of a multimodal dialogue and how frames are coded as messages in CHAMELEON. Worked examples in both theory and practice were given. Also described were our future intentions for a more complex active blackboard whereas at present the blackboard is simply a database of frames (messages). Next, we showed how CHAMELEON's dialogue manager manages synchronisation of inputs and outputs and sending/receiving frames between modules. At present the dialogue manager is implemented as a top-down controller but in future we will test other means of management which are completely distributed and even those in between. Finally, we discussed how DACS is used to glue the modules of CHAMELEON together.

Chapter 4

Domain model

The domain model contains all relevant domain data for the Campus Information System. This includes information on buildings, rooms and their functions and a list of all persons who work at the premises. Furthermore, a path planner is provided, helping planning of manoeuvres in the area by finding the shortest path between any two locations.

4.1 The physical environment

The campus is spread over a wide area, and although the current version of the system only covers a part of it, the implementation is prepared for an extension to include the whole campus. This is done by structuring the domain model in a hierarchical fashion, where we find *areas* at the top level. An area describes a complex of several buildings. The current area, Fredrik Bajers Vej 7, hosts our "Institute for Electronic Systems", but in general several institutes can be found in an area. At the next level of the hierarchy we find *buildings*, typically hosting an *institute* or *department*, but these can be spread into more buildings as well as one building hosting more than one department. At the third level, the bottom of the hierarchy, we find individual *rooms*. Each room in a building is described and categorised according to its type, e.g. *office*, *laboratory* or *corridor*. Each entity in the hierarchy has its own unique name, and its geographical location is recorded by coordinates, in the form of minimal (x-min, y-min) and maximal (x-max, y-max) x and y coordinates. These coordinates set the border of the (rectangular) area such that any pair of coordinates x, y is within this area if x-min $\leq$ x $\leq$ x-max and y-min $\leq$ y $\leq$ y-max. This is an idealisation as all entities are assumed rectangular. In reality this is not quite true, but we have approximated non-regularities in order to simplify the model and the associated search procedures.

Figure 4.1 shows an excerpt of the file containing the description of the physical environment. Levels in the hierarchy are made visible through indentation. The top level is further identified by the keyword *area*, and the description on the remaining line gives the name of the area followed by minimal and maximal x and y coordinates. The next level is identified by the keyword *building*, which is again followed by name and border coordinates.

```
area FRB7  0 1190 0 920
   building environment 0 1190 0 920

   building A1-1 871 1111 497 739
        tp      A1-1s1  725     898     2 A1-2s1 A2-1c1-1
   building A1-2 317 557 497 739
        tp      A1-2s1  725     341     2 A1-1s1 A2-2c1-1
   building A2-1 631  871 497 739
        room    00      624     663     746     860     meeting_room
        tp      A2-100  653     797     1 A2-1c2
        room    01      683     739     787     860     laboratory
        tp      A2-101  693     797     1 A2-1c2
        room    02      663     739     693     787     laboratory
        tp      A2-102  706     746     1 A2-102-2
        tp      A2-102-2 673    746     3 A2-1c2 A2-102 A2-102-4
        tp      A2-102-3 693    704     2 A2-103 A2-102-4
        tp      A2-102-4 673    704     3 A2-102-2 A2-102-3 A2-102-5
        tp      A2-102-5 640    704     3 A2-105 A2-102-4 A2-1c3-1
```

Figure 4.1: Excerpt of physical environment data

Within buildings we have two types of information. Rooms, which follow the same pattern with the addition of the type of the room, and trafficpoints (tp) that describe the points between which movement is possible. The first trafficpoint in each room (or building) has a special status - it is the origin of a route departing from the room. Each trafficpoint has a name, a coordinate pair and a number of neighbouring trafficpoints denoted by their name and preceded by (redundant) information on the number of neighbourpoints.

Building A2 is the one we model in detail and the name A2-1 refers to the first (ground) floor of this building. Each room in the area is referred as, for example, A2-102, and this reference is simply broken down to building A2-1 and the room name 02. Rooms with several entrances possess a trafficpoint for each entrance and possibly a number of intermediate trafficpoints to enable movement inside the room. Corridors especially have several trafficpoints each of them connected to several neighbourpoints in the different directions in which movement is possible. Building floors A1-1 and A2-2 are included as routes stretching over more than one floor past staircases in these buildings and the `building environment` is included in order to enable search functions to return reasonable answers, c.f. Section 4.3 below. Figure 4.2 displays the internal representation of the model corresponding to this information[1]. At system initialisation this structure is established and linked up. Arrows symbolise pointers through which we are able to access lower levels

[1] These figures do not precisely mirror the implementation where the internal structures are built up in reverse order. We have modified the figures slightly for convenience and clarity.

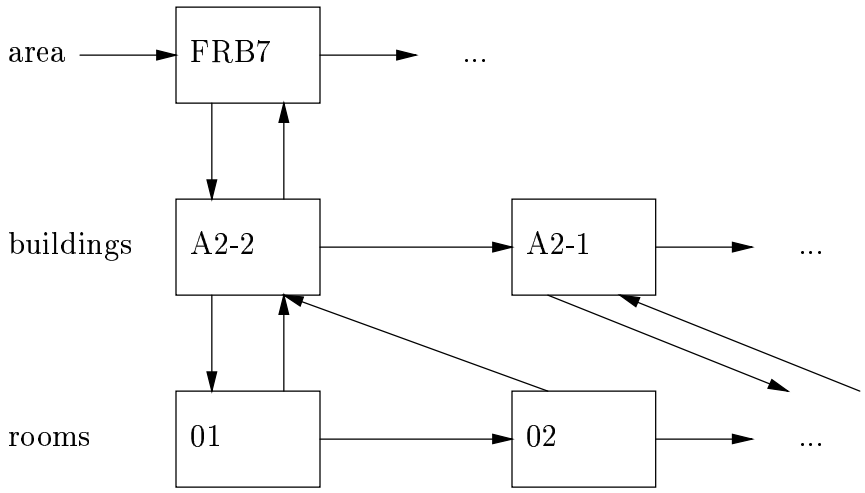

Figure 4.2: Internal representation of physical model

in the hierarchy or escape to higher levels.

Having described the physical structure of the campus we wish to extend with information about the functionality of each entity. For areas this could include information on faculties and institutes, for buildings it could be information on departments, laboratories or research groups, and for individual rooms it would typically include the tenants of offices and names of laboratories, etc. Only the latter is included in the current version of the system, but in order to enrich the dialogue with the system it would be obvious to extend the model along the sketched lines.

4.2 The people

All persons working in the campus and laboratories are registered in a separate file as shown in Figure 4.3. Persons are identified by the keyword *person* followed by information on their first names and surnames, their initials and title, departmental belongings and their offices given by area, building and office names. Likewise specific locations, most often referred to by name, are given by keyword *location* followed by the name and the room they occupy. Based on this information we build an internal representation as illustrated in Figure 4.4.

The reason for keeping this information in a separate structure is that we often wish to obtain facts of, for example, a specific person based on attributes of that person, most typically their name. Moreover, this part of the model tends to be much more dynamic than the hierarchy described above, as people move to new offices or staff arrive and depart. Locations appear in the personlist in this representation as their role is analogue to the persons.

```
person "Ove"    "Andersen"   oa   Civilingenioer          KOM  FRB7 A2-2 02
person "Jesper" "Jensen"     jje  Forskningsassistent     KOM  FRB7 A2-2 01
person "Claus"  "Nielsen"    cn   Forskningsassistent     KOM  FRB7 A2-2 01
 .
 .
location "meeting room"                              FRB7    A2-1 00
location "laboratory for speech coding"              FRB7    A2-1 01
```

Figure 4.3: Excerpt of people and locations data

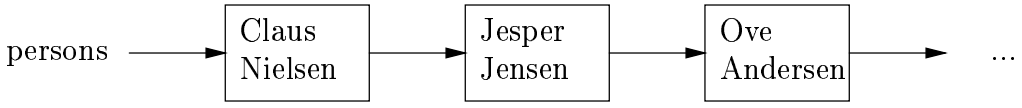

Figure 4.4: Internal representation of people and locations

To ensure fast responses when operating in the model, we link the structures together as shown in Figure 4.5. This enables us to retrieve room information from a person key by following a single link and vice versa. The pointers at the bottom of the personlist link people that share an office.

4.3 Functionality

The functionality of the domain model is to answer questions. Typical questions seek the office of a person, the tenant of a room or the room at a specific location. Thus, the functionality provided by the domain model is mainly search functions. Possible search keys are names of locations and persons (first, second or both names) and coordinates. The retrieved information consists of person or room information, the number of inhabitants in a room or the type of a room. The range of search functions is easily extensible as new functionality is required or new attributes are included in the model.

In order to cover every point in the 2D plans we have added dummy entities, for example the `building environment`, which covers a total area, as mentioned in Section 4.1. This is to be returned as a default if no building occupies the given coordinates.

In the current demonstrator names are internally transformed to unique identifiers by the NLP module. Ambiguities are currently handled by choosing the first possibility, if, for example, an ambiguous name (e.g. “Paul”) is entered. This simplifies the interface to the domain model as it always receives unique identifiers as arguments and always returns simple data items. Internally the domain model is prepared for more general interaction, however, with a cascade of search functions such as,

```
get_person_by_person_name(first_name, second_name)
```

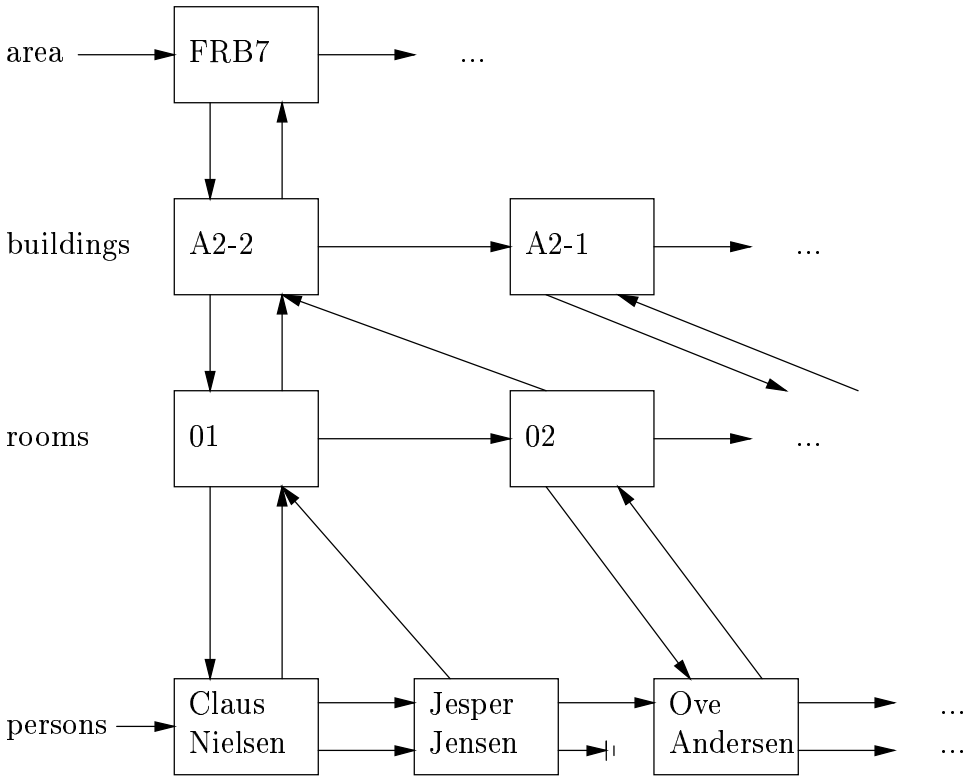

Figure 4.5: Complete linked internal structure

The current system makes use of the functions,

```
dm_get_person_coordinates(person_id)
dm_get_place_coordinates(place_id)
```

which results in a struct holding x and y coordinates,

```
dm_get_person_name(person_id)
dm_get_place_name(place_id)
```

that return a string with the requested name, and

```
dm_get_name_from_coordinates(xy)
dm_get_room_type_from_coordinates(xy)
```

that also result in a string being returned. For completeness, we list the additional functions, of which most are called by public functions or in the construction of internal data structures:

```
get_area_by_area_name(area_name)
get_building_by_building_name(area, building_name)
get_room_by_room_name(building, room_name)

get_area_by_coordinates(coordinates)
get_building_by_coordinates(area, coordinates)
get_room_by_coordinates(building, coordinates)

get_room_by_person_name(first_name, second_name)
get_room_by_person_id(id)

get_person_name_by_person_id(id)
get_person_by_room_name(building, room_name)
get_person_by_coordinates(coordinates)
```

that all return pointers to the corresponding type, and

```
get_no_of_inmates(room)
```

that results in an integer.

Consider, for example, a situation where we wish to find the name of a person at coordinates x and y, corresponding to the query "Whose office is this?" When the corresponding frame is analysed the dialogue manager finds the coordinates in the frame and realises that a name is required. This information is retrieved by the function call,

```
dm_get_name_from_coordinates(xy)
```

where xy is a structure with entries for the x and y coordinates. This function first identifies the room by the (internal) call,

```
get_person_by_coordinates(xy)
```

which takes advantage of the structure of the model as it searches according to the location hierarchy, by first finding the area, then the building, where the search is limited to buildings in the area, and finally the room, again limited to the rooms in the identified building. The latter function returns a pointer to the person attached to the room just found and a NULL pointer if no such person exists. Through this pointer the former function can get the number of tenants and retrieve the person's name which is returned as a string. If there are more tenants in the room the current version simply returns the name of the first one, but in general a list should be returned.

4.4 The path planner

In order to help navigation in the campus a path planner is provided, giving the shortest path between any two rooms. The rooms can be specified by coordinates, by name or function, as in "the speech lab", or by inhabitant as in "Lars Bo's office". From this specification we obtain information for the room by the search functions,

```
dm_get_person_tp(person_id)
dm_get_place_tp(place_id)
```

which returns a pointer to a trafficpoint. These functions use the search functions mentioned above and their results are used in the call of the path planner,

```
dm_route(source, destination)
```

that returns the path as a list of trafficpoint coordinates.

Trafficpoints are scattered throughout the model forming a graph, whose edges represent piecewise linear routes, along which passage is possible. The path planning is performed in this web by Dijkstra's shortest path algorithm (Aho et al. 1983). Dijkstra's algorithm works by setting the distance in the source of the route to 0, marks the node as visited and then visits all neighbours of this trafficpoint. The neighbours have their distance updated to the distance between the source and the neighbour node. This distance is computed as the length of the path between the two points, a simple calculation as this path is always a straight line. Now a new source is chosen as the node with the shortest distance, among those not yet marked as visited. The distances of its neighbours are updated to the distance to the current source plus the distance between the two nodes and the process proceeds recursively until all nodes in the web have been visited. During this process the shortest path is marked in the web and the result of the path planner is presented as,

```
route(coordinates(696,623,717,623,717,603,717,570,696,570))
```

where the coordinates are auxiliary x and y coordinates, each pair (x,y) describing a point along the path. At present the algorithm and implementation does not map subpaths running on the same line into full paths between the end points.

4.5 Implementation

The domain model is implemented in C. It could be argued that a common database program would suffice, but we have preferred to do our own implementation in order to be able to structure the data and optimise the algorithms freely. The program is quite straightforward, static information is held in simple text files as shown in Figures 4.1 and 4.3, which are read in at initialisation where an internal structure is build up linking information from the various parts of the database as shown in Figure 4.5. The module is compiled separately, and can be linked into any program that wishes to benefit from the provided functionality.

4.6 Future extensions

First of all an extension to cover the whole campus should be mentioned. This is a trivial, but quite tedious task, as coordinates for each and every room in the campus should be measured and typed in. If such an extension is decided, it should be considered to generate this information automatically from architectural drawings of the buildings. However, other information has to be generated manually. As indicated above there are several ways the domain model can be extended as the dialogue with the overall system is enriched. Information on faculties, institutes, departments and research groups are easily added, as is other information relevant for the extended dialogue.

One point of specific interest is an extension of the path planner to take advantage of the hierarchical structure of the locations. It seems possible to operate (more or less) independently on the different levels in the hierarchy. This could reduce the overall complexity of the task considerably, but the idea has not yet been pursued.

4.7 Summary

Here the domain model for CHAMELEON has been presented. The model contains all relevant data for the Campus Information System which includes data on buildings, rooms and their functions, and a list of all persons who work at the premises. A path planner plans manoeuvres in the area by finding the shortest path between locations. While most of the data in the domain model is obviously domain specific the path planning and data structuring aspects could be applied in other domains. Future work will involve augmenting the path planner to take advantage of the hierarchical structure of locations and also to investigate automatic techniques for knowledge elicitation to scale up the model possibly using the Topsy module (see Chapter 8).

Chapter 5

Gesture

When people communicate with each other face to face they use more than just speech. Body gestures and facial expressions are used to emphasize intentions and meanings. An intelligent user interface must be able to understand these phenomena. Gestures are one of the main inputs and outputs for CHAMELEON. We now describe how CHAMELEON recognises pointing gestures through image processing and also how it outputs pointing gestures using a laser.

5.1 Gesture tracking

Since no "off the shelf" gesture recognition solutions exist they have to be constructed. We now describe the construction of a gesture recognition module for CHAMELEON. We want to construct a gesture recogniser which will be able to recognise in the first instance pointing gestures. More complex gestures such as signals, marking an area or showing directions will be tackled in a later prototype. The gesture module can be used in the IntelliMedia WorkBench application where a user's intention can be recognised. For example, the intention might be to use a pointing device for pointing at an object (or whatever), indicating focus of attention. It is a requirement of the system that the user can point from any direction meaning that (s)he can stand where (s)he likes during an interaction. It is also a requirement that a typical pointing device such as a wooden stick can be used.

When the system is initialised a calibration module analyses the scene on the physical table of the WorkBench and the user cannot interact with the system in this period (5 seconds). The gesture module uses image analysis algorithms to recognise pointing gestures which provides a more natural interface since the user is not dependent on using special hardware such as data-gloves, touch screens or touch-sensitive surfaces. The pointing device is currently a typical wooden stick but in the future it will be replaced by an even more natural pointing device: human fingers/hands.

A static camera mounted in the ceiling above the WorkBench continuously captures images which are digitised by a framegrabber. From each digitised image the background

is subtracted leaving only object motions (and some noise) within the image. This motion is analysed in order to find the direction of the pointing device and its tip. By temporally segmenting these two parameters a clear indication of the position the user is pointing to at a given time is found.

In Figure 5.1 a flowchart of the overall algorithm of the gesture module is shown. Note that a calibration block and a mapping block (transform data) are present in the flowchart. The job of the transform data block is to make sure that the output of the gesture module is mapped to a format known by the rest of the system. The calibration block calculates the transformations used to do the mapping.

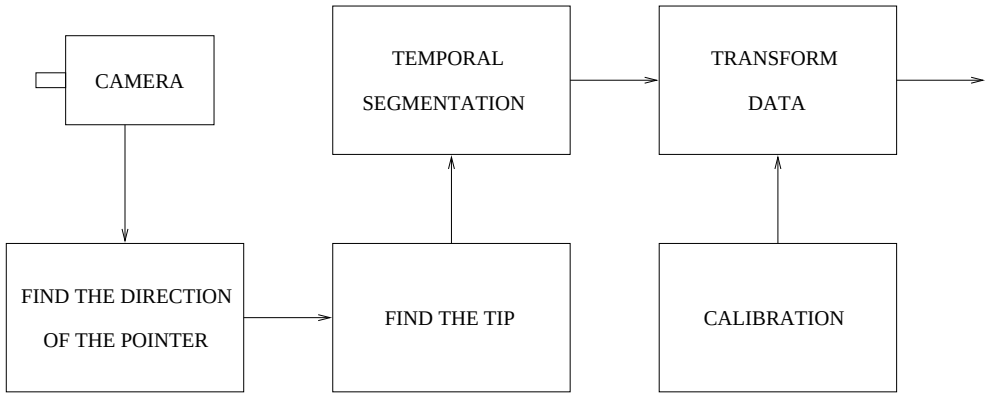

Figure 5.1: Overall algorithm

5.1.1 Gesture tracker

Figure 5.2 shows the physical table (black) with the 2D architectural plan (white) on top and three different areas are marked. The first (area 1) contains the outer boundary of the table and everything outside the table. The second (area 2) defines a band around the plan and the third (area 3) defines a region inside the plan. Area 1 is completely ignored by the gesture recogniser as is the area between the inner boundary of area 2 and the boundary of area 3. Therefore only data with respect to areas 2 and 3 are dealt with.

The algorithm runs as follows. First, area 2 (black) is searched for the pointer (white) direction and then this direction is used to search area 3 for the pointer tip (black). If no plan (or whatever) is present in the scene then area 2 and 3 and the space between them could be merged since they would be the same (black). In that case one search would be necessary (white on black)! But when a plan (white) is present there is a need for two different searches: pointer direction (white on black) and pointer tip (black on white).

Finding the pointer direction

When the system is initialised no pointer is present and the first image obtained will be stored as the background image. When a new image enters the gesture recogniser it is

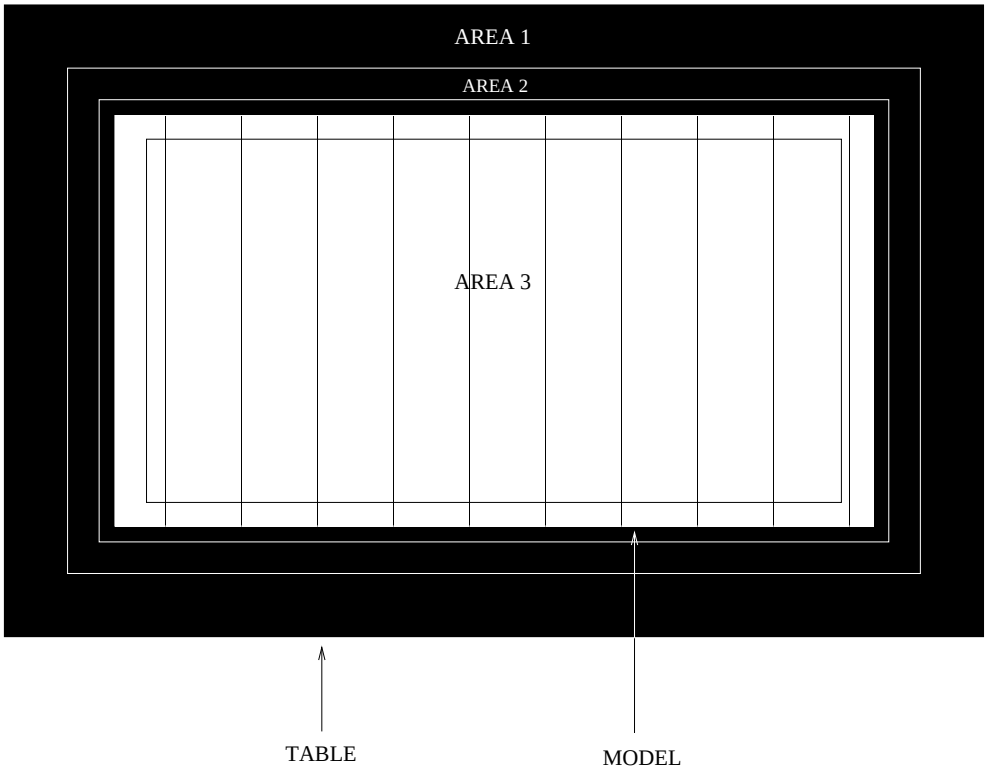

Figure 5.2: Different areas in the physical table

subtracted from the background image. Since the background is fairly black and the body of a pointer is fairly white the subtraction yields a huge value (100+) in area 2 where the pointer is present and a small value (20-) where no pointer is present. The data obtained from this subtraction is processed by a linear regression algorithm (Ross 1987) and the result is an equation which represents the pointer with respect to the image coordinate system. Some problems arise when the pointer is vertical since this yields an infinite slope in the equation. The problems are solved by setting the slope, the alpha value of the line, to 10, which is "close to infinity", and then recalculating the beta value of the line.

Whenever a pointer equation is found it is processed by an error control algorithm in order to detect errors. An error is defined as a situation where the points, which are believed to represent the pointer, and the equation of the pointer do not coincide approximately. A measure of this error is given by calculating the sum of squared differences (SSD) between the points found and the ones given by the equation. Say that the parameters of the line found by the regression algorithm are called α and β. Then the horizontal SSD, S_h, which is used when the pointer enters area 2 from the left or right side, can be defined as

$$S_h = \sum_{x=x_0}^{N_x+x_0} (Y_c(x) - Y_m(x))^2 \tag{5.1}$$

$$Y_c(x) = \alpha \cdot x + \beta \tag{5.2}$$

where N_x is the distance in the x-direction between the inner and outer boundary of area 2, x_0 is the x-coordinate of either the upper left corner of the outer boundary or the upper right corner of the inner boundary depending on whether the interaction is from the left or the right side, $Y_c(x)$ is the calculated y-value at a given x-position and $Y_m(x)$ is the measured y-value at a given x-position.

The vertical SSD, S_v, which is used when the pointer enters area 2 from the top or the bottom, can be defined as

$$S_v = \sum_{y=y_0}^{N_y+y_0} (X_c(y) - X_m(y))^2 \tag{5.3}$$

$$X_c(y) = \frac{y-\beta}{\alpha} \tag{5.4}$$

where N_y is the distance in the y-direction between the inner and outer boundary of area 2, y_0 is the y-coordinate of either the upper left corner of the outer boundary or the lower left corner of the inner boundary depending on whether the interaction is done from the top or bottom, $X_c(y)$ is the calculated x-value at a given y-position and $X_m(y)$ is the measured x-value at a given y-position.

If the result of the SSD is too large it either means that some noise is present in the scene (area 2) such as part of the user's body or some other object or the entire set up has been moved since the system was calibrated. In any case it means that an error is present and the current data is being processed no further.

Finding the pointer tip

How the tip of the pointer is found depends on what is placed on the physical table. If nothing is present on the table the body of the pointer (white) is used since it is easy to detect on the table background (black). The equation which was found in the previous search is now used to define the search area for finding the tip of the pointer (see Figure 5.3). The search is carried out in lines perpendicular to the line given by the equation. This is done as long as the pointer can be found or in other words until the tip is found.

When the plan is present on the table another technique is used. The equation which was found in the previous search is used again to define the search area for finding the tip of the pointer. Since the tip of the pointer has a very low intensity (black) compared to the plan (white) it is in general easy to use a method detecting the tip itself as the point having the lowest intensity in the search area. This is however not always the case due to shadows which can come from, for example, someone standing too close to the table. Hence, an additional algorithm is added to compensate for these. The problem is the following. If the user holds the pointer up high with respect to the table then the darkest pixel will be located where the pointer enters area 3 and not at the pointer tip. This can be solved by a second method saying that the pointer tip is found as the "dark-pixel" furthest away from where the pointer enters area 3. A "dark-pixel" is defined as

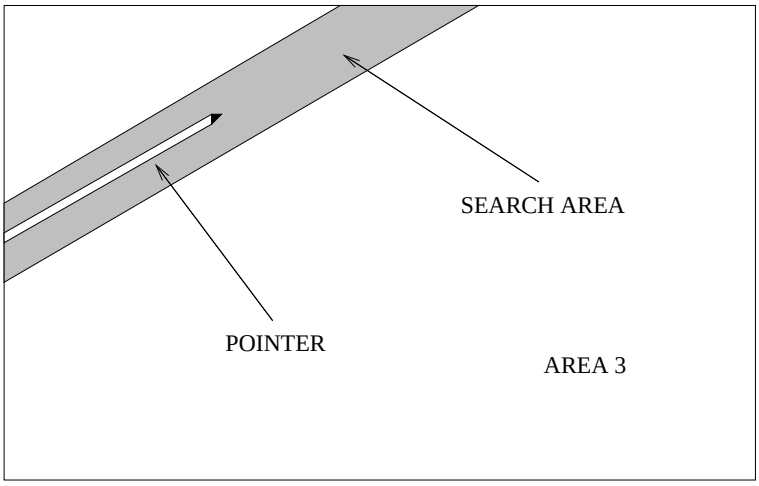

Figure 5.3: Pointer tip search area

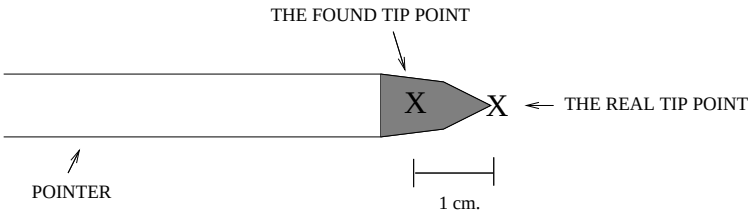

Figure 5.4: Extrapolating to determine the pointer tip

one having a graylevel value much darker (defined by a threshold) than the background. The problem with this solution is however, that it will sometimes find the pointer tip at the border of area 3 opposite the location where the pointer enters area 3. This happens whenever a spectator causes a shadow to enter area 3 at such a location. The solution to this tricky problem is to combine the two methods of finding the (1) absolute low-intensity point and (2) furthest away "dark-pixel" in the following way. Area 3 is divided into 2 regions with respect to the location where the pointer enters the scene. In the first region, from where the pointer enters and to the middle of area 3, the pointer tip is found as the "dark-pixel" furthest away and in the second region, from the middle of area 3 and to the outer boundary, the pointer tip is found as the darkest pixel. Tests have shown that the candidate tip determined by this method is located approximately 1cm from the real tip. Therefore the tip point is extrapolated 1cm in the direction given by the equation of the pointer. Now the true tip point is found (see Figure 5.4).

The new tip point (X_n, Y_n) is defined with respect to the old point (X_o, Y_o) as:

$$X_n = X_o + \Delta X \tag{5.5}$$
$$Y_n = Y_o + \Delta Y \tag{5.6}$$

with the relations between ΔX and ΔY given as:

$$(\Delta X)^2 + (\Delta Y)^2 = 1 \tag{5.7}$$
$$\frac{\Delta Y}{\Delta X} = \alpha \tag{5.8}$$

where α is the slope of the line indicating the direction of the pointer.
Isolating ΔY in 5.8 and inserting it in 5.7 yields:

$$(\Delta X)^2 + (\Delta X \cdot \alpha)^2 = 1 \quad \Leftrightarrow \tag{5.9}$$
$$(\Delta X)^2 + (\Delta X)^2 \cdot (\alpha)^2 = 1 \quad \Leftrightarrow \tag{5.10}$$
$$(1 + \alpha^2) \cdot (\Delta X)^2 = 1 \quad \Leftrightarrow \tag{5.11}$$
$$(\Delta X)^2 = \frac{1}{1 + \alpha^2} \quad \Rightarrow \tag{5.12}$$
$$\Delta X = \pm \frac{1}{\sqrt{1 + \alpha^2}} \tag{5.13}$$

where the sign of ΔX is determined by the direction of the pointing device.
Based on this result and equation 5.8 ΔY can now be calculated as:

$$\Delta Y = \Delta X \cdot \alpha \quad \Rightarrow \tag{5.14}$$
$$\Delta Y = \alpha \cdot \frac{1}{\sqrt{1 + \alpha^2}} \tag{5.15}$$

Note that the sign of ΔY is determined by the sign of α.

5.1.2 Temporal segmentation of tracker output

The gesture tracker component tracks the pointer in every frame. Since presently the other modules in the IntelliMedia WorkBench are only interested in whether or not a pointing gesture is present the output from the tracker must be filtered. Hence, the tracker will only give an output whenever it believes that a person is pointing and it will then output a set of 2D coordinates (x, y) and a timestamp (t). This will only happen one time for each position meaning that even though the pointer is held in the same position for some time the tracker will only give an output once.

The temporal segmentation is done by saying that the pointer has to be in the *same* position for δ frames. *Same* is defined as a distance with respect to a given position. The choice of this distance is not critical but has to be set with respect to the resolution in the image and the size of a room/office in the plan. When choosing δ several things most be taking into consideration. If δ is chosen too small the tracker will detect too many pointing gestures and if δ is chosen too large the user will have to hold the pointer still for a relatively *long* period of time in order to get pointing accepted. Tests have shown that

the user holds the pointer still for at least 1/2 sec. Since the frame rate is approximately 10 Hz then $\delta = 5$.

An example of a sequence being processed is shown in Table 5.1 below. From the table it is clear there is a delay of

$$(\delta - 1) \cdot (\frac{1}{10} Hz) = 400\ msec. \tag{5.16}$$

from when the pointing gesture is started until it is accepted. To compensate for this 400msec. are deducted yielding the correct timestamp.

Frame Number	*Timestamp msec.*	*Position X*	*Y*	*'Same' (position)*	*Output X*	*Y*	*Timestamp*
1	0	5.6	10.0	No			
2	100	5.7	20.2	No			
3	200	5.5	20.3	No			
4	300	5.7	20.1	No			
5	400	5.6	20.0	No			
6	500	5.5	20.0	Yes	5.5	20.0	100
7	600	5.6	20.2	Yes			
8	700	5.6	20.0	Yes			
9	800	15.7	30.1	No			
10	900	20.5	35.0	No			
11	1000	25.5	49.9	No			
12	1100	25.7	50.2	No			
13	1200	25.7	50.1	No			
14	1300	25.6	50.0	No			
15	1400	25.7	50.0	Yes	25.7	50.0	1000
16	1500	25.8	50.1	Yes			
17	1600	25.8	50.2	Yes			
18	1700	25.7	50.1	Yes			

Table 5.1: Sequence of tracker output

5.1.3 Calibration

In CHAMELEON and the IntelliMedia WorkBench application a camera and laser are intended to attend to physical objects. In the Campus Information System domain the physical object is a 2D plan of a building. When different sensors/actuators are working on the same physical system their internal representation of this system must be aligned or at least a transformation between the representations must be known. Since it is impossible to physically align the static camera coordinate system with the 2D plan a transformation

from the camera to the plan must be calculated. A calibration routine is designed in order to derive this transformation and more details are included in Appendix E.

Camera calibration

The camera uses a standard Cartesian coordinate system to mark an image with the origin in the upper left corner. The 2D architectural plan also uses this coordinate system which means that the transformation between the two systems is, in theory, linear and can be expressed as the product of a rotation matrix and a translation matrix (Gonzales and Woods 1993). By guessing one of the parameters in the resultant 3 by 3 matrix we end up with eight unknowns. To solve this matrix equation ($\mathbf{aX=b}$) we need four points (x, y) in the plan coordinate system and the corresponding four points in the image coordinate system.

The points in the plan are measured using a ruler with respect to the coordinate system having the origin in the upper left corner. Since the 2D plan is a rigid object these measurements only have to be done once.

Finding the image points

Every time the 2D plan and/or the camera are moved with respect to each other they must be recalibrated. The camera will not be moved very often but the 2D plan might be. Hence, the calibration should be done automatically.

The four image points should be easily located landmarks in order to make the calibration robust. We use the corners of the plan since they are present anyway and then no artificial markers have to be introduced. The points are found by searching for corner structures in the region between areas 2 and 3 in Figure 5.2 using information about the size and colour (B/W) of the 2D plan. Details on how this is done are found in Appendix E.

The system will now be self-calibrating with respect to the camera and 2D plan. This is done every time the gesture tracker is started up and lasts for about 5 seconds. Of course nothing can interact with the model in this period.

Uncertainty in the calibration

The uncertainty of the entire gesture recognition module depends mainly on the uncertainty in the calibration routine. Therefore some tests are carried out to evaluate the routine.

In the image the dimensions of the plan are 180x140 pixels which yields a resolution of approximately 0.6 cm/pixels in both directions. This is a rather coarse resolution but necessary since a huge area must be covered. 20 points are measured representing the entire area in the 2D plan and their locations are found in the image. The image points are then mapped into 2D coordinates and compared with the measured 2D coordinates. The average error is 0.442cm and no error is above 1cm. This means that most of the calculated errors are within one pixel and no errors are above 2 pixels. On the pixel level this is a fairly good result and since the pixel error is directly proportional to the error in the 2D plan this should be acceptable too.

The errors originate from two sources (the framegrabber might also generate some errors). First, the calibration method is based on the assumption that the camera is linear and can be modelled as a pinhole model (Gonzales and Woods 1993). However a camera is actually a very nonlinear device. To get a lower uncertainty a more advanced model from, for example, Tsai (1987) should be used. It is however concluded that the results obtained by the simple camera model are precise enough for the Campus Information System domain at least. Second, some errors might have been introduced when manually finding the points in the image but these errors will also be present when the system itself is responsible for finding the corresponding image points.

Transforming the data

Now that the transformation matrix has been computed, the temporal segmented data can be mapped into 2D plan coordinates using the following equation:

$$\mathbf{i} \cdot \mathbf{H} = \mathbf{m} \tag{5.17}$$

where $\mathbf{i}$ is an image point, $\mathbf{H}$ the transformation matrix and $\mathbf{m}$ the point in the 2D plan corresponding to the image point.

5.2 Laser system

In every interface between a human and a machine it should be possible for the machine to give feedback to the user in the most appropriate manner possible. With the IntelliMedia WorkBench we need a new kind of interface for giving feedback to the user so that the system as well as the user can point. A laser is used to give such feedback. A side effect is that the dependency between the user and traditional modalities (e.g. keyboard, mouse and screen) is released. Here we are concerned with the description and design of a laser system module.

A laser device is mounted in the ceiling next to the static camera as shown in Figure 2.3, Chapter 2. This device, in conjunction with the speech synthesiser, is used by the system to give responses to the user. The laser's response is given (1) as a *point* in response to user utterances like, for example, "Point to Thomas' office" or (2) as a *path* in response to user utterances like, for example, "Show me the route from Thomas' office to the computer room".

5.2.1 Laser device

We are using a red laser from *Laser Interface* (Lausen 1997) and a schematic representation of the laser is shown in Figure 5.5. Since the laser is used to draw points and paths at different locations it must be movable. This is managed by moving two small mirrors which reflect the laser beam. The laser beam is generated in the laser diode and the more current

this diode receives the more power the laser beam has. This is known as the modulation of the laser. The diode is a red (640nm) laser capable of generating 15 mW at 100% modulation. It is small (1cm) and cheap (500-1000$) and lenses are added to compensate for a lack in focus. After the beam has been focused it hits the two small mirrors (5x10mm) the position of which determine the direction of the the laser beam.

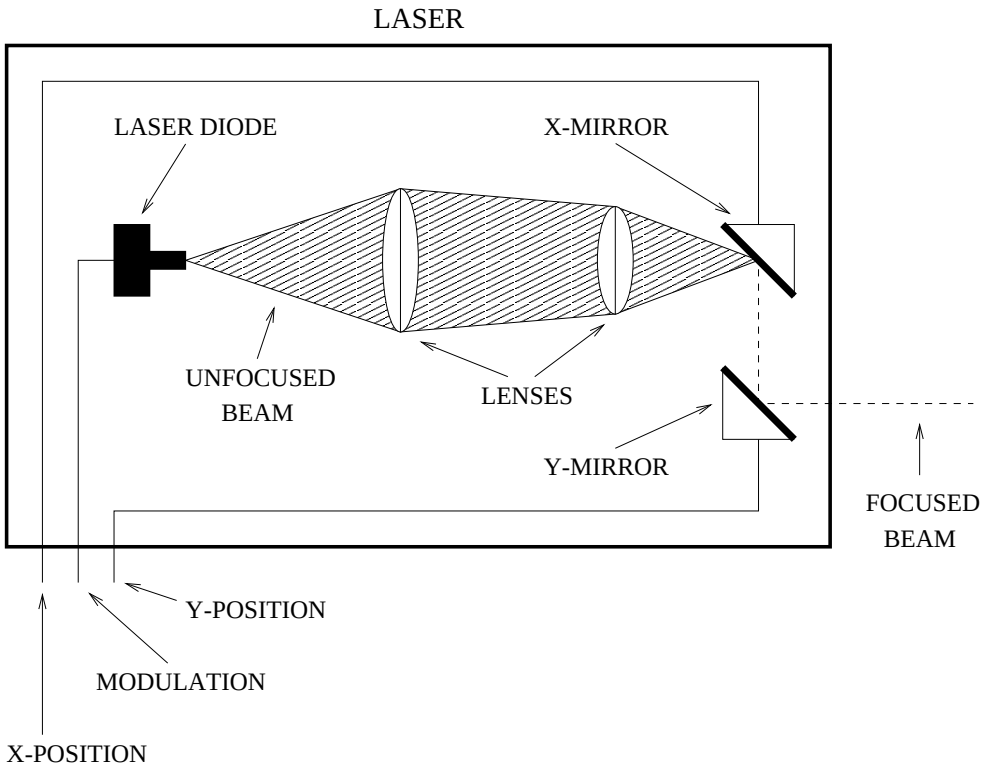

Figure 5.5: The internal structure of the laser

The control of modulation and mirrors is extremely fast. One can scan 600 points at 50 Hz which corresponds to drawing a path with 100 corners and updating it without the human eye noticing it. However, the more points, the lesser light per point meaning that not too many points should be present at the same time since this will make it harder to see the laser beam. It is not possible to solve this problem by just buying a more powerful laser because the laser's power spectrum moves into the infrared area as the power increases. This can be solved by buying a green laser but then the price increases tenfold. Also, it is not possible to just turn off/down the surrounding light source to increase the visibility of the laser since this will affect the quality of the camera's images. All in all the chosen laser is a fairly good compromise for the application. More documentation on the laser can be found in Lausen (1997).

5.2.2 Laser control

The laser is controlled from a dedicated Intel 486 PC. The PC is in turn controlled by another 200 MHz Intel pentium computer (r2d2) hosting the program requiring the laser. The laser and the controlling PC are not presently designed to give feedback which results in one way communication. This master/slave structure can be seen in Figure 5.6 where the arrows indicate one way communication. For further details see Moeslund et al. (1998).

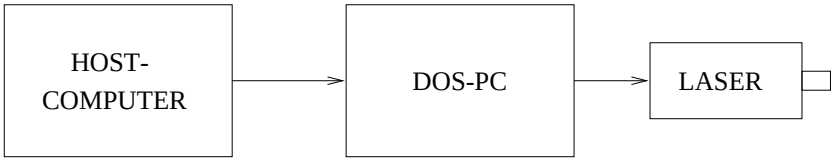

Figure 5.6: Master/slave control of the laser

5.2.3 Calibration

Before the laser can be used by CHAMELEON it must be calibrated to the 2D plan as was the case for the camera (see Section 5.1.3). The direction of the beam which is sent from the laser corresponds to the angles of the two mirrors. There is a linear dependency between the different angles meaning that the output will correspond to Cartesian coordinates if the laser is projected onto the inside of a sphere. This is however not the case since the laser is projected onto a flat table. Therefore the coordinates used by the laser will be distorted depending on the distance to the point of focus on the table with respect to a standard coordinate system and a correction algorithm must be applied (Moeslund et al. 1998). This correction algorithm transforms the coordinate system into a standard Cartesian system which can be calibrated in the same way as the camera (see Section 5.1.3).

There is a strong desire to make the laser calibration automatic so the user shouldn't have to worry about it. However, in order to do so a sensor needs to be involved and neither the laser or the 2D plan fulfill this requirement. The camera is a sensor and Figure 5.7 shows how the camera can be used to make the calibration automatic. In Figure 5.7 three transformations are shown. Transformation 2 between the laser and the 2D plan is the one needed. This transformation can be obtained by combining transformations 1 and 3. Since transformation 3 contains a sensor, the camera, it can be calibrated automatically. The transformation between the image and the 2D plan (1) is described in the gesture module. Transformation 3 is obtained exactly as 1 is by finding four points in one coordinate system and the corresponding four points in the other coordinate system. This is done by sending out four points from the laser at known positions in the laser's coordinate system. These points will then be picked up by the camera and the calibration can be carried out.

Tests have shown that the average error of transformation 2 is increased by 3mm by using this indirect calibration. This is however a small price to pay for automating the process. Transformation 3 has only to be recalculated whenever either the camera or the laser is moved which is very rare.

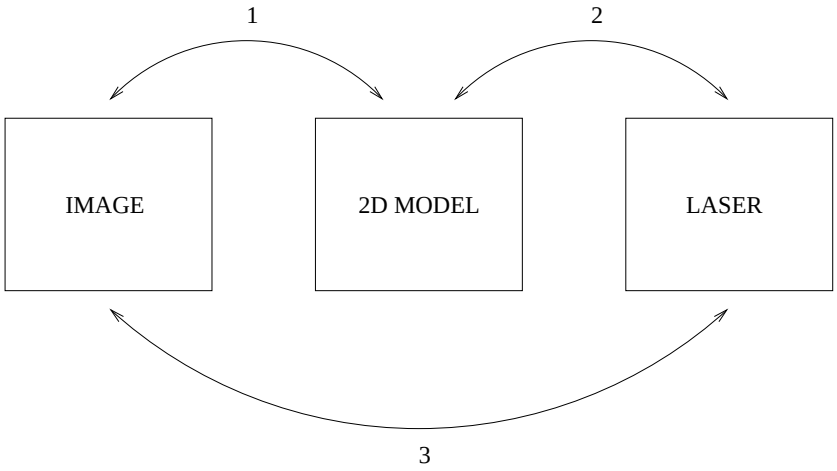

Figure 5.7: Coordinate transformations in the IntelliMedia WorkBench

To test that the different calibrations actually work a demonstrator which uses the gesture and laser modules is designed. The program finds the tip of the pointer and sends out the laser beam at this position so both transformations 1 and 2 are involved. The laser tracking demonstrator works successfully and clearly illustrates the effect of the calibrations.

5.3 Summary

Here, we discussed how gestures are implemented within CHAMELEON. There are two types of gestures: (1) those from the user and (2) those from CHAMELEON. At present CHAMELEON only recognises pointing gestures. CHAMELEON produces 2D coordinates accurate to 1 pixel for anywhere the user points to on the 2D model using a pointing stick. CHAMELEON uses a laser system to point and to draw paths between points. The laser is a red one which is less expensive and hence looses intensity when more is asked of it. Both gesture modules can be tested by asking the laser to follow the pointer tip. Future work will involve modelling more complex gestures such as marking-an-area or showing-a-direction.

Chapter 6

Speech

Spoken dialogue processing is one of the major components of any system which attempts to model language and vision processing and integration. Here, we discuss speech synthesis and recognition for CHAMELEON. We have bought and customised these devices and do not at present use the latest CPK speech recogniser (Christensen et al. 1998a,b) or synthesiser (Jensen et al. 1998, Nielsen et al. 1997) which are under development. We intend to experiment with these in later prototypes of CHAMELEON and also to possibly include some CPK work on emotions (Engberg et al. 1997).

6.1 Speech recogniser

This section describes the speech recognition module within CHAMELEON and the IntelliMedia WorkBench. It is introduced by a short discussion of the requirements both in terms of specific demands for supported languages, vocabulary size, interface to other modules and also in more general terms, such as speaking style and speech detection. The chosen speech recogniser is then described in accordance with the requirements.

6.1.1 Requirements

The overall requirement of the recogniser is that:

- The user must be able to address the system in a way that he/she finds intuitively natural.

This requirement can be broken down in a number of more specific requirements:

(1) The recogniser must be able to handle continuous speech.

(2) No pre-training must be required before a new user can speak to the system, i.e. the recogniser must be speaker-independent.

(3) The recogniser must be able to handle spontaneous speech phenomena such as hesitations, false starts (restarts) and throat sounds. Robustness against environmental noise and other speakers must also be provided.

(4) Furthermore, a specific vocabulary and grammar must not be enforced upon the user, at least not in a manner which conflicts with what he/she intuitively would express him/herself.

The fact is that very few systems are able to fulfill all four requirements although most cope with (1) and (2) and some also with (3) and (4) to some degree.

6.1.2 Specifications

In addition to the features described above a set of specifications more related to implementational issues is defined:

(1) The recogniser must run in realtime meaning that the recognition results must be available within a very short period (1-2 seconds) after the user has stopped speaking.

(2) The recogniser must be able to accept at least the English language but preferably also Danish and others.

(3) It must be possible to interrupt the system when speaking. Speech detection must be carried out implicitly, i.e. for example the user must not be forced to press a button to speak.

(4) The recogniser must be able to run with the hardware and software chosen for the implementation of CHAMELEON which in the present case are Intel PCs with Linux or SUN Sparcstations with Solaris.

(5) The recogniser must be compliant with the formats chosen for intercommunication between the modules in CHAMELEON within the DACS IPC system.

(6) The recogniser must have a well-documented Application Program Interface (API) and be compliant with Hidden Markov Models (HMM) and grammar formats supported by the Center for PersonKommunikation (CPK).

6.1.3 Choice of speech recogniser

In order to comply with the requirements and specifications listed above we have chosen to use the grapHvite speech recogniser (version 1.0) (Power et al. 1997, Odell et al. 1997), produced by Entropic Research, for a number of reasons.

First, the recogniser supports the Hidden Markov Model ToolKit (HTK) (Young et al. 1996) standard for Hidden Markov Models (HMMs) and grammar network which is widely used and available at the CPK. Second, it contains pretrained speech models

for British and US English, German, and Spanish together with transcribed lexicons for up to 100k word(forms). Furthermore, Danish speech models available at CPK can be used directly within grapHvite. grapHvite also offers a graphical environment for defining grammar networks. Our NLP module parser described in Chapter 7 is also able to generate the finite state grammar network format used by grapHvite from the parser's compound feature grammar. This ensures compatibility of the grammar used in the NLP and speech recognition modules of CHAMELEON. Third, grapHvite is well documented and supports a number of platforms including C and JAVA APIs.

6.1.4 Configuration, vocabulary and grammars

A very demanding set of specifications were made for the recogniser in Section 6.1.2. These are briefly revisited and discussed with regard to the chosen grapHvite speech recogniser.

Graphvite supports near realtime continuous speech recognition. However, spontaneous phenomena are only handled to a certain degree, and for example hesitations and user interrupts are not supported at a sufficiently robust level. A noise-cancelling microphone is used that minimizes the influence from noise and other speakers.

In order to allow the user to express him/herself freely only a rudimentary grammar has been defined corresponding to the previously defined set of instructions/commands that the system is able to process (see Chapter 2, Section 2.2). The recogniser will only identify phrases carrying significant information within the user's utterances. The phrases are denoted speech concepts and the recogniser can be said to spot concepts rather than to attempt full recognition of user utterances. This approach has the advantage that only a limited grammar and vocabulary has to be defined covering the speech concepts thus eliminating the need for extensive language modelling. The risk is, of course, that recognition becomes less robust for complex user utterances as the system only has a very limited linguistic competence.

For the present the only supported language is (British) English. However, the domain of application of the IntelliMedia WorkBench is inherently bilingual as Danish person names are a significant part of the vocabulary. Hence, there is a need for at least two languages being available simultaneously. This is not a problem as the recogniser can work with a number of different models simultaneously and these can in turn be from different languages.

6.1.5 Speech recognition example

This section shows and discusses a trace of the recognition of a spoken command to the IntelliMedia WorkBench. As described in Section 6.1.3, the grapHvite speech recogniser includes a graphical environment for creation and evaluation of the application grammar and vocabulary. The grammar is represented as a RTN (Recursive Transition Network) which is converted into a finite state network by the recogniser. Figure 6.1 shows the topmost part of the network.

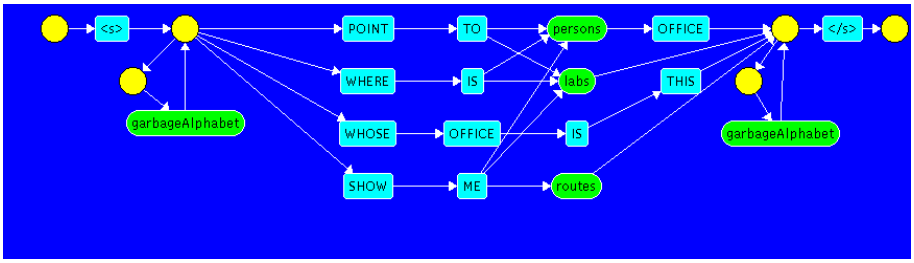

Figure 6.1: Snapshot RTN from grapHvite network tool

Figure 6.2 shows a trace of the speech recogniser and parser processing the spoken utterance: "Show me the route from Tom's office to the speech lab" The first part (1 to 19) shows the progress of the speech recogniser in 50 ms steps. The symbols in '<>' denote either silence <s> or a phoneme model number <02>. The phoneme models are used to discard unknown words, and thus enable the recogniser to distinguish (spot) content words within for example a longer sentence. Note that the recognition process stops when a period of silence is encountered (19).

21 shows the final string, converted to lower case and with ',' delimiting the individual words. A timestamp is added and the string is put into a frame which is sent to the NLP module (24). The NLP module builds syntactic and semantic representations of the utterance (see Chapter 7) and note that its semantic frame (33) has substituted more specific identifiers for the locations with referents ("tb" and "a2_102"). The timestamp remains unchanged however, as the time refers to that of the user's spoken utterance and not the time of parsing.

6.2 Speech synthesiser

Speech output is a necessity in CHAMELEON for addressing the user with instructions, queries, and declarative information in the right tone of voice. This can be achieved by generating pure synthetic speech through, for example, applying text-to-speech (TTS) synthesis directly to a text string. Another method is to use prerecorded human speech. The speech is recorded as single words or short phrases and then concatenated into sentences and played back to the user. This method will under some circumstances sound more natural than synthetic speech but has the problem of being very inflexible. For example, if new words/phrases have to be added the person who was recorded originally has to participate in all subsequent recordings which is often problematic. Furthermore, the more pieces that have to be patched together the more unnatural the resulting sound will be.

TTS on the other hand sounds more unnatural but has the advantage of flexibility since it is not tied to a specific vocabulary but generated dynamically "on the spot" when needed. Therefore, a TTS synthesiser is employed in CHAMELEON where a variety of

```
1:      @50 <s> <02>
2:      @100
3:      @50 <s>
4:      @100 <s>
5:      @50
6:      @100
8:      @150 <s>
9:      @200
10:     @50
11:     @100 <s> <23> SHOW
12:     @150 <s> <23> SHOW ME JORGEN
13:     @200 <s> <23> SHOW ME THE ROUTE
14:     @250 <s> <23> SHOW ME THE ROUTE FROM TOMS
15:     @300 <s> <23> SHOW ME THE ROUTE FROM TOMS
16:     @350 <s> <23> SHOW ME THE ROUTE FROM TOMS OFFICE TO THE
17:     @400 <s> <23> SHOW ME THE ROUTE FROM TOMS OFFICE TO THE
18:     @450 <s> <23> SHOW ME THE ROUTE FROM TOMS OFFICE TO THE SPEECH_LAB
19:     @500 <s> <23> SHOW ME THE ROUTE FROM TOMS OFFICE TO THE SPEECH_LAB <16> <s>
20:
21:     show,me,the,route,from,toms,office,to,the,speech_lab
22:
23:
24:     getting message 7:
       speech_recognizer(utterance(show,me,the,route,from,
       toms,office,to,the,speech_lab),time(890048013))
25:
26:     SYNTACTIC TREE NO 1
27:     ....
28:     ....
29:             (a trace of the NLP parser output is omitted. The reader is
                referred to Chapter 7 for a detailed discussion of the
                parsing process)
30:     ....
31:     ....
32:     SEMANTIC TREE NO 1
33:     sending message nlp(intention(instruction(show_route)),
       source(location(person(tb),type(office))),
       destination(location(place(a2_102))),time(890048013))
34:
```

Figure 6.2: Trace from speech recogniser and NLP modules

spoken output is anticipated.

6.2.1 Synthesiser requirements

A number of basic requirements for the TTS synthesiser have been defined:

(1) The synthetic voice must be intelligible and natural sounding.

(2) It must be possible to manipulate the voice with regard to several issues, most notably the speed, volume and pitch.

(3) It should be robust against common abbreviations and support direct transcription instead of plain text. This is necessary in order to get correct pronunciation of special words or phrases. For example, "1998" might be pronounced differently depending on the intended meaning (see Section 6.2.2 below) and "Clausewitz" would not be expected to be in a generic pronunciation lexicon.

(4) It should support English and Danish and preferably simultaneously so that they can be mixed (e.g. for Danish names like Børge in English utterances).

(5) It should have a well-defined Application Program Interface (API) and a simple interface to the communication formats and protocols within CHAMELEON.

(6) It should be possible to closely monitor and control the playback of the synthesised sentence.

The first two requirements refer to voice quality and the remaining requirements relate to synthesiser flexibility. For example, it is important that common abbreviations such as amounts ("kr." = kroner), dates ("Jan." = January), and titles ("frk." = frøken (miss)) can be expanded directly by the synthesiser so texts do not have to be preprocessed.

6.2.2 Choice of synthesiser device

The INFOVOX 700 TTS synthesiser (Infovox 1994) fulfills a number of the above requirements. The synthesiser is a self-contained device complete with rechargeable battery, power supply, and loud speaker. It communicates via a standard RS-232 serial connection which makes it highly flexible and easily attachable to any host architecture. INFOVOX provides simultaneous support of multiple languages which is very convenient for our purposes where we must mix Danish proper name pronunciation into English utterances. Unfortunately, the voice quality is not acceptable in the long term but for our initial application prototype it is adequate. A number of very convenient features include:

Multiple languages: The present version supports Danish, (British) English, (Castillian) Spanish and Swedish. Language switch within sentences is possible, thus enabling the synthesiser to, for example, pronounce a Danish name correct within an otherwise English sentence.

Speaking rate: For experienced users and/or long passages it might be convenient to increase the speaking rate.

Speaker style: The pitch of the voice can be changed from "deep male" to "child" giving a darker or lighter quality to the voice and changing the aspiration. The voice can be made more or less expressive by changing the intonation (pitch dynamics) settings. Also, the volume can be set to a desired level. The sonority of the voice can be changed from whispering to booming. A number of voice settings have been predefined.

Speaking mode: The speaking mode of the synthesiser can be set to *spelling*, *word-by-word* or *sentence*. For example, spelling is convenient if a word such as a proper name is mispronounced. The system can then spell it to the user. If a specific, non-standard pronunciation is required, the synthesiser can enforce a user-supplied transcription, thus overruling the default transcription rules. This is convenient for unusual or foreign words.

Direct control over prosodic features: This is important, when for example trying to disambiguate a user utterance. As an example take: "Do you mean Paul *Mc Kevitt's* or Paul *Dalsgaard's* office?" with stressed words emphasized. Here stress is used to point the user's attention to the two surnames which he/she is being asked to disambiguate.

Figure 6.3 below (cf. Infovox 1994, p. 1-5) shows the processing steps from a text string to the speech signal.

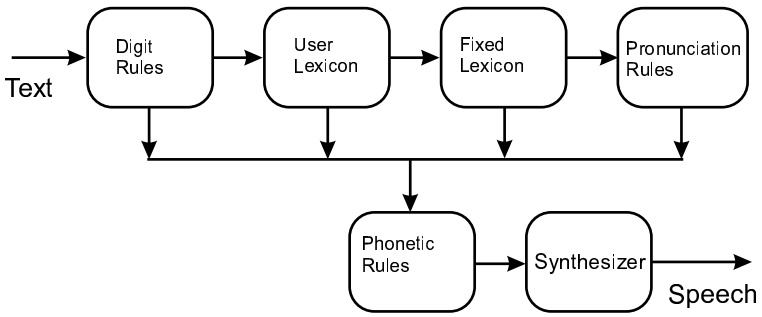

Figure 6.3: Processing within the INFOVOX 700 TTS synthesiser (cf. Infovox 1994, p. 1-5)

The top row of processing blocks are preprocessing steps from the "raw" text string into a phonetic representation. *Digit Rules* convert all digit strings in a word into appropriate formats. For example, "1998" is pronounced "nineteen hundred and ninety eight" which is the default rule for pronunciation of years. If the user has over-ruled this by specifying

numeric mode ($1998) it will be pronounced "one thousand nine hundred and ninety eight". The user can specify (force) the pronunciation of selected words by including them in the *User Lexicon*, which takes precedence over the *Fixed lexicon*. If a word cannot be located by either, it is submitted to the language-specific, built-in pronunciation rules.

When the preprocessing of the text string is accomplished, the string is passed on to the *Phonetic Rules* which determine a large number of parameters for the synthesiser. For example, this includes the duration and stress of each phoneme and the sentence intonation contour and actual sound quality. Phenomena such as coarticulation and reductions are also handled. Finally, the string is transferred to the synthesiser and the actual speech sounds are generated.

The requirements (2-5) in Section 6.2.1 can be fully met. Requirement (6) is unusual for a TTS device and is based on the fact that the synthesiser is used in a MultiModal context and must be closely synchronised with other modules (e.g. the laser system). For example in the utterance "This is Paul Dalsgaard's office and this is Paul Mc Kevitt's office" the synthesiser must say Paul Dalsgaard at the same time as it is pointing at the coordinates of his office. Hence, it is a requirement that information about the exact timing of every word as well as the remaining time must be available.

6.3 Summary

Here we have discussed the requirements for the speech recognition module and features of the chosen grapHvite speech recogniser. Some of the requirements cannot be met, i.e. those mostly to do with robustness and grammar/vocabulary coverage. Most notably, support of HTK formats, multiple languages and a well documented API (Application Program Interface) led to the choice of grapHvite.

The chosen INFOVOX 700 speech synthesiser has proven to be highly flexible and was easily integrated into the system. It provides many useful features which can be controlled through a simple, text based format. It is capable of speaking a number of languages with varying speaking styles and voices. A drawback is that the voice quality is too poor in the larger perspective and this is language specific, as the quality of Swedish and (British) English are especially higher. However, for demonstration and prototyping purposes, this is not considered a major problem.

Chapter 7

Natural language processing (NLP)

The natural language processing (NLP) module, which is still under development, consists of a number of separate submodules the core of which are the *natural language parser* and the *grammar converter*. The parser takes output from a speech recogniser and returns semantic frame representations. The converter converts the grammar used by the parser into a language model (e.g. the HTK standard lattice format (Young et al. 1996)) to be used for constraining the search space of the speech recogniser. These two submodules function within a speech understanding system as shown in Figure 7.1 below.

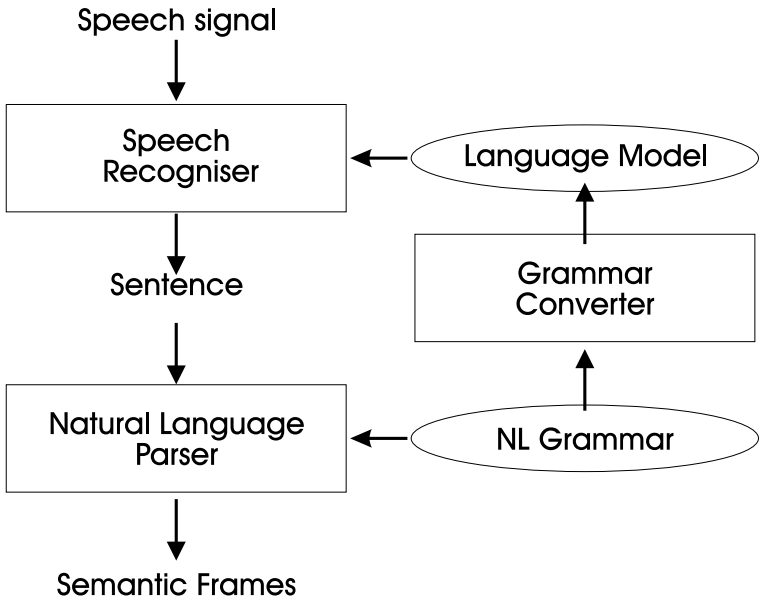

Figure 7.1: Speech understanding system

Further, submodules include a grammar constrained *typed text recogniser* that simulates speech recognition. The text recogniser takes typed text as input and returns another string

that is "similar" to the input and at the same time grammatical according to a predefined finite state *language model*. Hence, it can substitute the speech recogniser in Figure 7.1.

The NLP module is being developed partially within IntelliMedia 2000+ and partially within the REWARD EU-project (see http://www.kom.auc.dk/CPK/Reward and Bai et al. 1998). Parts of it are rooted in The Danish Spoken Language Dialogue Project (Bækgaard et al. 1995). Within IntelliMedia 2000+ the modules are integrated into the general CHAMELEON platform and the IntelliMedia WorkBench application whereas in REWARD they are to be interfaced to tools for designing and interpreting subgrammars and spoken dialogues (Bai et al. 1998). Here, we focus on the natural language parser (Section 7.1) and the grammar converter (Section 7.2), and some other submodules are briefly described.

7.1 Natural language parser

The parser is a unification-based left-corner chart parser which processes input bottom up with top-down filtering (left corner dependencies) and left-to-right. The parsing algorithm largely corresponds to the enhancement of Earley's algorithm (Earley 1970) presented by Shieber (Shieber 1985) and the "Earley-algorithm" implemented in the NJAL parser (Music 1994). It is implemented in C++ and includes a standard ANSI C API (Application Program Interface). The API mainly provides commands for loading external grammar files, activating and deactivating subgrammars (if the grammar is organised into subgrammars) and, of course, for parsing.

The parsing function takes the 1-Best recognition result from a speech recogniser and returns a pointer to a C structure containing zero or more semantic trees (nested semantic frames). The argument of the parsing function can be an orthographic representation or a phonetic transcription of a recognised sentence. The phonetic transcription is a sequence of, for example, SAMPA (Speech Assessment Methodologies Phonetic Alphabet) or CMU labels (as used for example in the TIMIT (Texas Instruments & MIT) speech database, HTK transcription files) denoting the acoustic monophone or triphone models matching the speech signal. If parsing fails, the returned C structure contains no semantic trees, in case of ambiguity two or more trees, otherwise only one tree. A structure containing one tree may be considered the "normal" case (parsing successful, no ambiguity). If parsing fails and the parser resorts to recovery strategies the C structure may contain frame fragments instead of frame structures denoting different "meanings" of the entire utterance. This is then indicated by a flag in the C structure. Alternative parsing functions taking N-Best lists and lattices in the HTK Standard Lattice Format as arguments (Young et al. 1996, p. 197) are yet to be implemented.

For the latest version of the parser see WWW: http://www.kom.auc.dk/~tb. The parser source code is publically available for research in accordance with the GNU General Public License. The parser can be compiled and run using any C++ compiler under any Operating System tested so far: GNU c++, Sun CC, Borland bcc32, Sun Solaris, Windows 95, and Linux. In short, the parser has been implemented in 32 bit C++ standard code

and the ANSI C API interface enables interfacing to programs implemented in principally any programming language: Prolog, Lisp, Java and so on.

Currently, the parser supports three external grammar formalisms: (1) The ICM (Interpretation and Control Module) Grammar Format (Bækgaard et al. 1992, p. 9 ff.), a label-based (non-unification based) recursive transition network format which until recently has been used for speech understanding and recognition in restricted, system directed spoken dialogue systems at CPK; (2) a formalism based on a subset of the EUROTRA User Language developed originally for machine translation, however sufficient general to meet other NLP demands as well (Bech 1991, Maegaard 1985)[1]. CPK has experience in using such a subset for speech understanding from a strategic programme Spoken Language Dialogue Systems carried out in collaboration with CST (Centre for Language Technology, Copenhagen) and CCS (Centre for Cognitive Science, Roskilde) (cf. Brøndsted 1994, Music 1994, and Povlsen 1994). The EUROTRA subset formalism is henceforth denoted APSG (Augmented Phrase Structure Grammar), the "augmentation" consisting of the fact that the formalism is compound feature based and supports various binary and unary operators; (3) finally, also a trivial label-based context-free PSG (phrase structure grammar) or "BNF" format is supported.

The differentiation between external grammar (ICM, APSG, or PSG) and internal grammar representation (in the source code, a C++ object denoted *NLPgram*) is crucial in the design of the parsing modules. The fact that the internal representation is capable of storing different external formats such as ICMs and APSGs, and that the *grammar converter* described below is capable of converting the internally represented format into finite state networks in the form of HTK Standard Lattices, Vocalis' recognition grammar format (see Section 7.2) holds promise that many other popular NLP and recognition grammar formats can also be supported. For each externally supported grammar formalism, the parser implements a LEX/YACC (and Flex/Bison compatible) format definition to be compiled and linked with the C++ code. An overview of the architecture of the parser is shown in Figure 7.2. In parallel, for each supported recognition grammar format the converter implements a corresponding write procedure (see Figure 7.3, Section 7.2).

7.1.1 Test programs

For each implemented LEX/YACC parser, the parser has been compiled as a stand-alone test program using the API. The name of the test programs consist of a three letter code denoting the kind of external grammar being supported ("aps", "icm", "psg") followed by "test": apstest (compiled to read APSG grammars), icmtest (compiled to read RTN grammars in the ICM format), psgtest (compiled to read context-free PSG grammars). The three letter codes reoccur as extensions of the grammar demonstration files. The test programs take the name of an external grammar definition as first argument and the name of a file with test sentences as second argument (apstest mmui.aps mmui.snt,

[1] Unfortunately, the official EUROTRA documentation is difficult to get hold of. For the implementation of the APSG YACC parser, the designer has used non published working documents prepared at CST (Center for Sprogteknologi), Copenhagen, Denmark.

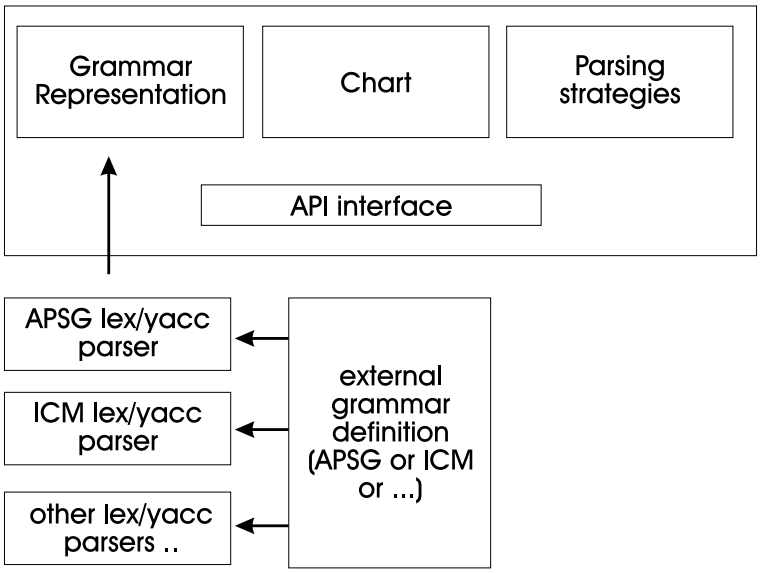

Figure 7.2: Architecture of parser

icmtest mmui.icm mmui.snt etc.). If the second argument is left out, the internal grammar representation is dumped to the screen and, of course, there will be no 1:1 relation to the external format. A grammar definition may contain more named subgrammars. By default, all subgrammars are activated by the stand-alone programs. Otherwise, third and subsequent arguments are interpreted as names of subgrammars to be activated. With options setting trace and debug levels, the test programs can be forced to display parse trees and an entire chart. By default, they only output the sentence being parsed and the resulting semantic frames.

7.1.2 From utterances to semantic frames

As explained above, the parser itself, unlike many, is kept strictly separated from the actual grammars used for parsing. Hence, the parser itself can be considered fairly theory independent. The grammars can be large and very general ones as well as small "toy" grammars designed for very restricted speech understanding tasks. The grammar designed for the Campus Information System (shown in Appendix F) belongs to the latter type, though some effort has been made to formalise more general syntactic rules. The grammar, written in APSG format, has been used to generate the semantic frames in the examples shown below. All tracing information such as dumps of parse trees, has been removed so that the list reflects nothing but the actual communication between speech recogniser, NLP, and dialogue manager modules via the blackboard.

```
(1) show me paul dalsgaards office
(English(nlp(intention(instruction(pointing)))
(location(person(pd))(type(office)))(time(887464117))))
```

```
(2) point to paul dalsgaards office
(English(nlp(intention(instruction(pointing)))
(location(person(pd))(type(office)))(time(887464117))))

(3) where is paul mckevitts office
(English(nlp(intention(query(where)))
(location(person(pmck))(type(office)))(time(887464117))))

(4) show me paul broendsteds office

(5) where is pauls office
(English(nlp(intention(query(where)))
(location(person(pd))(type(office)))(time(887464117))))
(English(nlp(intention(query(where)))
(location(person(pmck))(type(office)))(time(887464117))))

(6) show me instrument_repair
(English(nlp(intention(instruction(pointing)))
(location(place(a2_105)))(time(887464117))))

(7) point to the laboratory
(English(nlp(intention(instruction(pointing)))
(location(place(a2_101)))(time(887464117))))
(English(nlp(intention(instruction(pointing)))
(location(place(a2_102)))(time(887464117))))
(English(nlp(intention(instruction(pointing)))
(location(place(a2_103)))(time(887464117))))

(8) show me the route from toms office to the dialogue_lab
(English(nlp(intention(instruction(show_route)))
(source(location(person(tb))(type(office))))
(destination(location(place(a2_103))))(time(887464117))))

(9) zxcv
```

As seen here, some (most) utterances are unambiguous (1,2,3,6,8), some are ambiguous (5,7), and some do for some reason not make any sense at all (4,9). Some utterances are synonymous (1-2) or partly synonymous (3-5). In brief explanation we should add that there are two Pauls and three laboratories in the A2 building domain. The two Pauls are Paul Dalsgaard and Paul Mc Kevitt, and the three laboratories are speech, coding, and dialogue. There is no Paul Brøndsted but only a Tom Brøndsted in the A2 building. Finally, the word "zxcv" is unknown within the domain and by incident, we believe, also

in the rest of the world! This information is stored in the Campus Information System grammar which consequently must be considered very domain specific. For "historical" reasons, the input sentences are orthographic representations and not phonemic labels denoting the HMMs (Hidden Markov Models) matching the speech signal. As the *grammar converter* described in Section 7.2 was not ready for use as the Campus Information System was implemented, the language model used by the speech recogniser was generated by hand with grapHvite (Power et al. 1997) as described in Chapter 6 which presupposes orthographic representations.

In general terms, the semantic frames returned by the parser consist of zero or more nested predicate-argument structures. In the frames the uppermost predicate "English" is the name of the subgrammar that produced the subsequent frames (the name occurs in the beginning of the grammar file shown in Appendix F). This predicate is not inserted by any rule of the grammar, but is created by the parser as part of the API definition. In user-directed dialogue systems subgrammars may be useful in clarification subdialogues, or, if the system is multilingual, for preventing, for example, Danish rules from interfering with English ones. Otherwise, in extreme system-directed spoken dialogue systems, it makes sense to design one specialised subgrammar for each "system prompt" in the dialogue. As subgrammars are used also for constraining speech recognisers, this "trick" will help improving recognition.

The predicate "time" has an argument denoting the UNIX system time which has been calculated by a postprocessing function. Postprocessing functions are implemented using the API as described below. The fact that all time frames in the examples above have the same timestamp gives evidence that the sentences, admittedly(!), were not spoken and recognised by a speech recogniser but actually loaded from a file (and, of course, it gives evidence that the parser was able to parse all of them within a second!). Finally, we may draw attention to the predicates "place" and "person" which take arguments (e.g. pmck, pd, a2_101, a2_102) that are percolated from as far back as from the lexicon. The percolation of values through unification is explained below.

In general, the parser goes through two steps to generate a list of semantic frames from a sentence. First, it uses the axiomatic, lexical, and syntactic rules to create a parse tree describing the (general) syntactic structure of the sentence. The parse tree for the sentence "show me paul dalsgaards office" is shown below:

```
"s1_imp" line 195 (s){stype=imp }
[
   "vp_1" line 218 (vp){vform=imp }
    [
        line 165 (verb){lex=show,phon=sh ow sp,nb=$N, valence=np_np,
                                                        vform=imp,prs=$P }
      "np_pron" line 244 (np){case=acc,semtype=person,nb=sing, prs=$P }
      [
           line 175 (pron){case=$C=acc, lex=me, phon=m iysp,nb=$N=sing,
                                                  prs=$P=1, subtype=person }
      ]
```

```
        "np_gen" line 238 (np){case=no,semtype=place }
        [
            "np_person2" line 252 (np){case=gen,semtype=person,id=pd }
            [
                 line 70 (proper){case=no,semtype=person,surname=no,lex=paul,
                                                  phon=p ao l s|p oh l sp,id=pd }
                 line 73 (proper){ 1 case=$C=gen, semtype=person,surname=yes,
                                 lex=dalsgaards, phon=d ae l s g ao z sp,id=pd }
            ]
            "np_place2" line 261 (np){case=$C=no,semtype=$S=place,id=office,
                                                          nb=$N,prs=$P,def=no }
            [
                 line 154 (noun){ case=$C=no, semtype=place,lex=office,
                                       phon=oh f ih s sp,id=$I=office,nb=sing }
            ]
        ]
    ]
]
]
```

In a second step, the parser looks for semantic mapping rules associated with the subgrammar that produced the parse tree (the subgrammar "English"). If it doesn't find such rules, it will simply derive the semantic frames from the top node of the parse tree:

```
(English(stype(imp)))
```

This technique of semantic extraction presupposes extended percolation of values to the top node defined through syntactic rules and is not used in the Campus Information System. At the end of the grammar shown in Appendix F, there are a number of semantic mapping rules each of which expresses an action and a condition. The condition is a subtree which must match (unify with) a subtree in the parse tree in order that the action is carried out. The action results in semantic frames which may be linked together with other frames so that they, finally, describe the semantics of the entire sentence:

```
(English(nlp(intention(instruction(pointing)))
(location(person(pd))(type(office)))(time(887464117))))
```

The "semantics" of the sentence is, of course, application specific and not general. On the other hand, as opposed to the syntactic level (the "parse tree") we consider the frames relatively language independent, in spite of the "English" labels which could easily be substituted with German, French, or any arbitrary labels. A Danish grammar that, for sentences corresponding to the English ones, produces exactly the same semantic frames can easily be implemented, i.e.:

```
vis mig paul dalsgaards kontor
(Danish(nlp(intention(instruction(pointing)))
(location(person(pd))(type(office)))(time(887464117))))
```

In general, we think that it is possible to code grammars that on the syntactic level are relatively general although language dependent and on the semantic level are language independent although domain specific. The grammar coded for the Campus Information System doesn't fulfil this requirement since we have focussed more on the parser and integration with other modules.

7.1.3 APSG grammar formalism

The Campus Information System MMUI grammar, shown in Appendix F, is coded in the unification-based, compound feature based APSG format which to a large extent is a subset of the EUROTRA User Language used at many research institutes in Europe. Hence, we assume that the grammar is immediately accessible to many computational linguists. However, the APSG format is actually a little bit more than just a EUROTRA subset. It also implies some extensions or "novelties". We will briefly go through the different types of rules found in Appendix F, starting with the back bone of the entire formalism: the feature set.

Feature set

The core of the formalism is the feature set, that is a set of features enclosed in '{' '}' each of which consists of an attribute assigned to a value with the '=' operator. The features of a feature set are separated by the binary "and" operator ','. The feature set shown below is the lexical rule for "me", a personal pronoun in accusative, singular, first person.

```
{lex=me,phon="m iy sp",cat=pron,subtype=person,case=acc,nb=sing,prs=1}.
```

The funny feature `phon="m iy sp"`, is the phonetic transcription of "me" using the TIMIT set of phonetic symbols (Garofolo 1988, Garofolo et al. 1993). Unlike most other unification based formalisms, the APSG format only allows atomic and not complex (nested) features. In future versions of the parser, the internal representation of grammars will be extended to support complex features thus allowing parsing with, for example, the very general (theory-independent) PATR-II formalism (Shieber 1986). PATR-II theoretically allows coding of different linguistic frameworks such as GPSGs (Generalised Phrase Structure Grammars) (Gazdar et al. 1985, Pollard 1984), and HPSGs (Head Driven Phrase Structure Grammars) (Pollard 1984, Pollard and Sag 1994), and LFGs (Lexical Functional Grammars) (Bresnan 1982).

In the APSG formalism, every feature set must contain a "cat" feature. In unification based grammars, this feature is usually used for linguistic constituent names like s, np, vp, and pp (sentence, noun phrase, verb phrase, and prepositional phrase) and parts of speech names (POS names, word classes) n, v, adv, and pron (noun, verb, adverb, and pronoun).

The EUROTRA User Language allows definitions of feature declaration tables to prevent grammar designers from assigning other values to attributes than the ones declared for the attribute. Currently, the APSG formalism does not include this facility. On the other hand, this offers the advantage that the "cat" feature in application specific subgrammars can be used for anything which make sense within the restricted domain (pseudo linguistic categories like person, room, route, etc.)

There are three reasons, why the "cat" feature is obligatory. First, the basic operation performed by the parser on feature sets is unification (cf. Shieber 1984, 1985, 1986, Kay 1985). This means that two feature sets are unified to a new feature set if there is no explicit contradiction between them (if they do not assign different values to the same attribute). Obviously, if we do not insist on a common feature attribute being present in all feature sets, a slip of the pen, e.g. "catt" for "cat", can bring unification to "explode". For instance, the rule above unifies with the illegal feature set

```
{catt=np,case=$C,nb=$N}.
```

where we have changed "cat" to "catt" (note that `$C`, `$N` are variables which are instantiated during unification as described below). Secondly, the parsing strategy is based on the Earley chart parsing algorithm (Earley 1970) optimised with so-called top-down filtering (Shieber 1985), a prediction mechanism based on the cat-features which secures that only rules having a chance of building the relevant constituent structures are inserted into the chart. This makes parsing much more efficient although strictly not necessary. Finally, the cat feature is used as mutual structure for constituent and POS names in unification based grammars and for labels in label-based formalisms like the ICM format. In the internal representation of feature sets in the parser, cat-features are moved outside the set and the redundant attribute name "cat" is removed, e.g.

```
(pron){lex=me,phon="m iy sp",subtype=person,case=acc,nb=sing,prs=1}.
```

A label in, for example, the label-based ICM formalism is treated as an empty feature set, e.g.

```
(personal_pronoun){}.
```

If the parser displays parse trees, which can be forced with a trace option, the separate nodes of the tree are shown as cat-faced feature sets without regard to the external format. Further, the order of the features constituting the set may have changed, as the parser re sorts them for optimisation of unification.

The drawback of making the cat feature obligatory should not be minimised. For instance, the very efficient technique of postverbal subcategorisation described in Shieber (1986, p. 27 ff) presupposes that cat features can be variables. Future versions of the parser may annul the restriction concerning the cat attribute.

Macros, ad hocs

The defined APSG format allows definition of macros and so-called ad hoc feature sets. Both macros and ad hoc's are nothing but time-saving facilities that spare grammar designers some typing work. A macro has the form:

```
#n3sg = {cat=noun, number=singular, person=3}.
```

and are subsequently referred to like in the rule:

```
{lex=man, #n3sg, stype=hum}.
```

which internally is expanded to:

```
{lex=man, cat=noun, number=singular, person=3, stype=hum}.
```

"Ad hoc" rules are popular with designers of small subgrammars which do not attempt to comply with "linguistic correctness". They are of the form:

```
<from><to>
```

Such labels are expanded to:

```
{lex=from,cat=from}.
{lex=to,cat=to}.
```

Basically, ad hoc feature sets express that the grammar designer refers to "this word" and takes no interest in the linguistic properties of the word.

Types of rules

The APSG format includes four types of rules: (1) axioms, (2) structure building rules, (3) lexical rules, and (4) semantic mapping rules. For the parser, all types inherit from the same basic form: a rule consists of a left-hand-side which is a feature set, and a right-hand-side which is a network of feature sets (often a simple sequence of feature sets). Either side can be empty and the network depth can be greater than one. In the case of axioms, the left-hand-side is empty, in the case of lexical rules (l-rules) the right-hand-side is empty, in the case of structure building rules (b-rules) neither side is empty. Mapping rules (m-rules) are like b-rules, except the network depth can be greater than one and they contain information for building nested semantic frames.

All rules can be preceded by a label which serves only tracing purposes. When the parser displays parse trees, each node of the tree is labelled with the label identification (if any) and the line number of the completed rule that built the node.

Finally, all rules can be assigned to a subgrammar. An entry `%%##<sub %%grammar id>` in the APSG grammar file will assign a special subgrammar feature

{subgrid=<sub grammar id>} to all subsequent rules. Note that the first two characters "%%" of a subgrammar entry will "fake" a EUROTRA parser that knows nothing about "subgrammars". This is due to the fact that "%%" normally prefaces free comments. Global rules, with no subgrammar feature, can be defined in the beginning of the file, before the first subgrammar entry. Subgrammar features associated with rules participate in unification like normal features: two feature sets of two rules can be unified if the subgrammar feature of the rules do not contradict. This means that global rules can be unified with other global rules as well as with subgrammar rules. However, subgrammar rules can only be unified with other subgrammar rules if they belong to the same subgrammar.

Axioms

Axioms define the topmost categories or start symbols of parse trees. When no axiom is defined, the YACC parser defines a rule internally:

```
DEFAULT_AXIOM=[{cat=s}].
```

which corresponds to what most linguists expect the start symbol to be: a sentence! The edged brackets '[]' denote that the feature set(s) expressing a sequence. However, explicit definition of axioms is more flexible and allows, for example, isolated noun phrases to be analysed as what they are: noun phrases and not "elliptical sentences". This can be achieved with two axioms:

```
[{cat=s}].
[{cat=np}].
```

which optionally can be combined into one rule using the binary "or"-operator ';':

```
[{cat=s};{cat=np}].
```

Note that if ';' is replaced by the "and" operator ',', the parser will attempt to analyse input as a sentence followed by a noun phrase. Of course the feature sets of an axiom can include more than one feature, e.g.:

```
[{cat=s,stype=declarative}].
```

which will prevent the parser from recognising imperative sentences, questions, and so on. Temporary modifications of axioms are useful in connection with some of the debugging tools that come with the NLP module.

B-rules

B-rules define the constituent structure of sentences. Consider the rule:

```
%% e.g. show sbdy sth
vp_1 = {cat=vp, prs=$P, nb=$N,vform=$V}
   [ {cat=verb,prs=$P,nb=$N,vform=$V,valence=np_np},
     {cat=np, case=acc},
     {cat=np, case=acc}
   ].
```

The rule describes a verb which takes two nominal objects, e.g. "shows you something", "tell him something". The features for person and number and the verb form are percolated to the head from the left-most child of the body when this is unified with, e.g. the l-rule for "shows":

```
{lex=shows,phon="sh ow z sp",cat=verb,prs=3rd, nb=sing,valence=np_np}.
```

The result of such unification is a new rule:

```
vp_1 = {cat=vp, prs=3rd, nb=sing,vform=$V}
   [ {lex=shows, phon="sh ow z sp",cat=verb,prs=3rd,vform=$V,nb=sing,
                                                        valence=np_np},
     {cat=np, case=acc},
     {cat=np, case=acc}
   ].
```

where `lex=shows, ... nb=sing,valence=np_np` is the two unified feature sets and where `prs=3rd, nb=sing` are variables instantiated as a side effect of this unification. The idea of percolating values from the body to the head is, of course, to prevent the new rule from being combined with a subject not agreeing with its person and number features (*"the men shows ...", *"I shows ..."). The variable `$V` remains uninstantiated. In case the b-rule is instantiated with the imperative form of "show":

```
{lex=show,phon="sh ow sp",cat=verb,vform=imp,valence=np_np}.
```

`$V` is instantiated and `$P` and `$N` are uninstantiated. The feature `vform=imp` allows the rule to describe an imperative sentence with no subject ("show me something"! as opposed to *"shows me something"!).

Percolation of features up through the parse tree is achieved with local variables being present in both the head and the body of b-rules. Agreement (e.g. checking person-number agreement) is expressed with one variable being present twice or more in the body of a b-rule. The grammar in Appendix F uses combined percolation and agreement to check if first names and last names of the staff list fit together and to percolate the IDs of these persons to the head of the rule.

```
%%tom broendsted, tom broendsteds
np_person2 = {cat=np,case=$C,id=$I,semtype=person}
   [ {cat=proper,case=no,id=$I,surname=no},
     {cat=proper,case=$C,id=$I,surname=yes} ].
```

This implies that the parser rejects an input like "tom dalsgaard" as ungrammatical and interprets "paul" as ambiguous: "Paul Dalsgaard"?, "Paul Mc Kevitt"?. The motivation for this analysis is that the grammar is used not only for parsing but, in a derived form, also for speech recognition. Allowing non existing first and last name combinations would increase the perplexity of the derived language model and result in poorer speech recognition.

The coding of bodies (right-hand-sides) of b-rules and axioms may include all binary and unary operators defined for b-rules in the EUROTRA framework. The binary operators are ',' (AND), ';' (OR), and the unary operators are (empty transition, "jump", "skip"), '*' (zero or more iterations), and '+' (one or more iterations). To group feature sets that are object to such operators, the parentheses '(' , ')' can be used. A trivial noun phrase definition for English can be coded:

```
{cat=np,number=$N}
[
    (
     ^{cat=det},
     * {cat=adj},
     {cat=n,number=$N],
     ^{cat=pp}
   );
   {cat=pron,number=$N}
].
```

The rule enumerates (describes) phrases like "man", "the man", "the tall man", "tall heavy men", "tall heavy men in England", "he", "they", "I", "me", etc. Internally, such rules are reorganised as state transition networks consisting of a number of nodes and arcs, the latter being feature sets. The test programs described in Section 7.1.1 will, when invoked only with the name of the APSG grammar file as argument, display the rules as networks.

L-rules

Lexical rules are simply feature sets. Besides the special 'cat' feature discussed above, lexical rules must include at least one dedicated feature used for lexical look-up. The APSG grammar shown in Appendix F implements two such features "lex" and "phon" which store the orthographic and the phonetic transcription(s), respectively. The actual attribute used for lexical lookup is defined by calling a Boolean function:

```
BOOL SetLexKey(char *AttributeName, char Delimiter =' ')
```

which creates a lexical interface using the values of the attribute given in the first argument so that the value strings are divided into substrings at the character defined by the second argument (by default: space). Further, the parser will use the defined delimiter to separate input strings (sentences) into tokens (words). The function call, either SetLexKey("`lex`") or SetLexKey("`phon`",'`|`') is currently hardwired into the YACC parser but will in later versions be configurable (definable in the grammar files).

The function call SetLexKey("`phon`",'`|`') will for the two lexical "paul" rules in Appendix F create a lexical interface:

```
p ao l sp       {lex=paul,phon="p ao l s|p oh l sp",id=pd,cat=proper,
                                 case=no,semtype=person,surname=no}.
                {lex=paul,phon="p ao l s|p oh l sp",id=pmck,cat=proper,
                                 case=no,semtype=person,surname=no}.
p oh l sp       {lex=paul,phon="p ao l s|p oh l sp",id=pd,cat=proper,
                                 case=no,semtype=person,surname=no}.
                {lex=paul,phon="p ao l s|p oh l sp",id=pmck,cat=proper,
                                 case=no,semtype=person,surname=no}.
```

Subsequently, the parser will expect input strings to be of the form:

```
sh ow sp|m iy sp|p ao l sp|d ae l s g ao z sp| oh f ih s sp
```

(show me Paul Dalsgaards office) where tokens are separated at '`|`'. As the converter described in Section 7.2 uses the same source code for reading grammars and representing them internally, the SetLexKey function will also affect the derived recognition language models and ensure that the representation of the "sentences" returned by a speech recogniser using these language models agrees with the expected format. As explained above, the Campus Information System has been implemented using the "`lex`" features as lexical key.

M-rules

Semantic Mapping Rules (m-rules) do not have any direct counterpart in the EUROTRA User Language though there is some resemblance with the so-called "filter rules" (f-rules) of this framework. The M-rules is an enhancement of the rules originally developed for the Spoken Language Dialogue System mentioned above (Music 1994, Povlsen 1994). The enhancement consists in the fact that m-rules now not only create separate semantic frames but also link these frames into larger nested frame structures. Further, semantic frames can be post processed using the API (see Section 7.1.4).

As explained in section 7.1.2, a (sub)grammar needs not include m-rules. In the absence of m-rules, the parser derives semantic frames from the top node of syntactic parse trees. The first version of the Campus Information System Grammar in Appendix F was

implemented using no mapping rules. The fact that the APSG format, and the internal representation of grammars in the parser, only supports atomic, non-nested features, imposes too heavy restrictions on which frame structures can be created. This lead to the enhanced concept described below. However, with support for complex features which is to be implemented in future versions of the parser, the generation of semantic frames based on conventional syntactic parsing will have the same formal power as the concept described below. A method based on complex features will resemble the concept of generating "logical forms" of sentences described by (Shieber 1986, pp. 32).

M-rules consist of an action part and a condition part separated by a slash '/'. The condition part describes a syntactic subtree and resembles the b-rules except for the fact that their body (right-hand-side) can have a depth >=0. Further, the condition part may include link action information, a label preceded by # and replicated at least once in the action part. The action part consists of constant or variable semantic frames so that atomic frames, frames with no arguments, may be a replicated link action, e.g.

```
point1 =
(nlp
   (intention
      (instruction(pointing)),
    #LOCATION,
    time($T)
   )
)
/
{cat=s}
[
{cat=vp}
[
   {cat=verb,lex=point},
   {cat=pp}
   [
      {cat=prep,lex=to},
      {cat=np}#LOCATION
   ]
]
].
```

This rule states that if a syntactic parse tree includes a subtree unifying with the condition after '/', the frame structure defined in the action before '/' will be created and that "#LOCATION" will be substituted by frame structures previously created at the noun phrase (np) node. Each syntactic parse tree is scanned bottom up and each node of the tree is matched with each m-rule of the same subgrammar until unification succeeds. This means, that semantic mapping, as opposed to syntactic parsing, is not exhaustive but returns as soon as an action is carried out.

The semantic frames returned by the parser are normally the structure which is created at the top node of parse trees (the node build by an axiom) and which may have been linked to other structures created elsewhere in the tree. However, if the parser is not capable of creating semantic frames at this node, it will scan the parse tree top-down and copy frames found in daughter nodes to the C structure returned to the blackboard. In this case, a flag in the C structure will indicate that the list of nested frames does not contain alternative meanings but semantic frame fragments (see Section 7.1). The frames returned when parsing with the ICM grammar format are always fragmentary as the ICM format presupposes that the global semantic structures denoted "semantic representations" are described elsewhere in a dedicated Dialogue Description Language (DDL) formalism (Bækgaard et al. 1992).

For a description of post processing of semantic frames (e.g. the frame `time` in the m-rule above), we refer to the next section.

7.1.4 The API

The API (Application Program Interface) of the parser is implemented in ANSI C as opposed to the rest of the NLP module which is implemented primarily in C++ to enable linking to programs implemented in languages like Prolog, Lisp, Java, etc. The API mainly provides commands for loading external grammar files, activating and deactivating subgrammars, if the grammar is organised into subgrammars, and, of course, for parsing. The actual parsing function takes the 1-Best speech recognition result as argument (a NULL-terminated string) from a speech recogniser and returns a pointer to a C structure containing the nested semantic frames. As described in Section 7.1, later versions of the parser will include functions for parsing N-best lists and word lattices.

In the current version of CHAMELEON, the parser is linked directly to the speech recogniser bypassing the blackboard. This is due to the fact that the parser currently is the only device processing the output from the speech recogniser. Thus, the parser together with the speech recogniser constitute a new device, a "speech understanding device", which writes frames to the blackboard. Though, the parser and the speech recogniser are linked together statically, they communicate using messages (frames) defined by a BNF grammar (see Appendix D). The recogniser transforms the recognised utterance into a string denoting the frame structure (e.g. "show me paul dalsgaards office" into "speech_recognizer(utterance(show,me,paul,dalsgaards,office),time((887464117)") and then the parser analyses this string and creates the appropriate frame structure. The semantic frame returned by the "speech understanding device" to the blackboard does not include the subgrammar information (the uppermost argument "English", cf. Section 7.1.2) as it is not yet part of message (frame) syntax. A multilingual Danish-English version of the Campus Information System is being considered and in such a version the subgrammar information would be passed to the blackboard as, of course, English requests must be answered in English, Danish requests in Danish, etc.

The parser API also includes methods for implementing post processing functions. A post processing function is a function that takes a semantic frame as argument, with access

to all daughters of this frame, and modifies it. Typically, post processing functions may involve arithmetical operations for calculation of numerals, dates, and so on, e.g.,

```
year(hundreds(19),tens(9),ones(8)) > year(1900,90,8) > year(1998)
```

The Campus Information System grammar only implies one post processing function, namely the function that calculates the time frame. The implementation of a post processing function involves three steps: (1) declaration of the function in a certain declaration file (postfunc.def), (2) definition of the function in a file containing all post processing functions (postfunc.cc) and (3) recompilation of the parser (using the makefile that comes with the NLP module). The declaration consists of a placeholder `DEF_FUNC` for the type of function, a string defining the label of the frame to be postprocessed, and the actual name of the function, e.g.,

```
DEF_FUNC("time",calctime)
```

The definition is of the form:

```
void calctime(APIsemframe *s)
{
   ....
}
```

where `APIsemframe` is the ANSI C structure defined in "nlpapi.h". The function may delete existing labels (frame names), children, etc. using the dedicated functions defined in "nlpapi.h". A flag in each frame indicates if the label string is owned by something else than the frame itself. For instance, all labels created by the parser before post processing are owned by the grammar being parsed and cannot be de-allocated. Their lifetime lasts until the grammar is deleted, for example, until a new grammar file is loaded. Label strings owned by the frame itself as well as the actual frame structure tree exist until the parser starts parsing a new sentence. Label strings owned by the frame must be allocated using the standard ANSI C function "malloc" (not "new") as they are de-allocated with "free" (not "delete").

7.2 The grammar converter

In terms of source code, the parser and grammar converter overlap extensively. Basically, the converter simply replaces the parsing algorithm and chart with a number of converting functions and write procedures. The LEX/YACC routines for reading external grammars, the internal grammar representation, and a library of functions for unification, matching, etc. are the same as in the parser. The architecture of the converter is shown in Figure 7.3.

The purpose of the converter is to generate language models to be used for speech recognition and to ensure a degree of equivalency, in terms of linguistic coverage, between

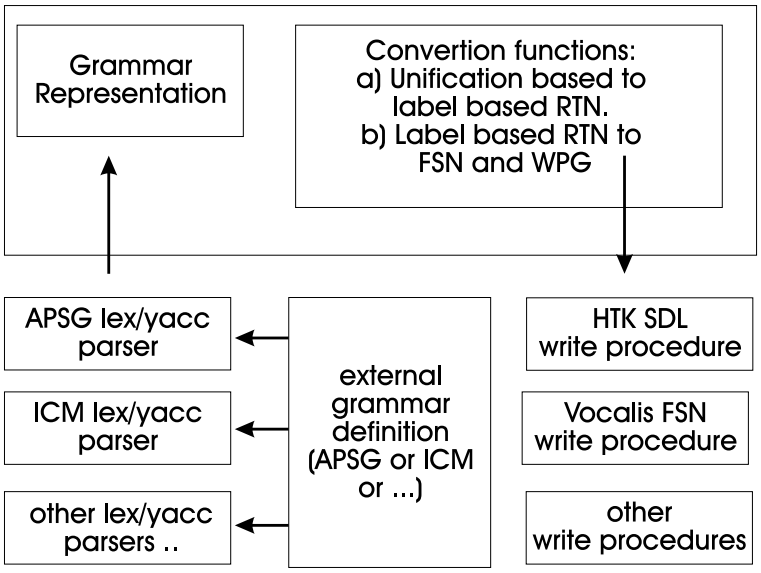

Figure 7.3: Grammar conversion architecture

the sublanguage being recognised and the sublanguage being parsed (cf. Figure 7.3). The finite state language models being generated by the converter are either weakly equivalent or approximately equivalent to the external grammar. As regards equivalency of grammars see Winograd (1983, pp. 112 ff.). Weak equivalence means that the external grammar and the derived finite state language models accept exactly the same set of sentences, however without assigning the same structures to the sentences (in which case they would be "strongly equivalent"). Approximate equivalency means that the set of sentences accepted by the finite state language model is somewhat larger than the set accepted by the external grammar.

7.2.1 Converting programs

For each implemented LEX/YACC parser, the grammar converter has been compiled as a separate program. The names of the converting programs consist of a three letter code denoting the kind of external grammar being converted ("aps", "icm", "psg") followed by "cnv": apscnv (compiled to convert APSG grammars), icmcnv (compiled to convert RTN grammars in the ICM format), psgcnv (compiled to convert context free PSG grammars). The converters take the name of an external grammar definition as first argument and the name of the file (actually files!, see below) to be written as second argument (apscnv mmui.aps mmuiALL, icmcnv mmui.icm mmuiALL etc.). By default, all subgrammars are activated by the converter. Otherwise, third and subsequent arguments are interpreted as names of subgrammars to be activated. The finite state networks being generated by the converting program encompass all active subgrammars. To generate one network for

each subgrammar, the appropriate converter has to be invoked for each of them: apscnv mmui.aps mmuiGB English, apscnv mmui.aps mmuiDK Danish etc. Note that the commands, apscnv mmui.aps mmuiALL will not mingle Danish word sequences with English ones. They are kept apart, but a speech recogniser supporting the subgrammar concept will not be able to determine if a recognised sentence is English or Danish, unless the two languages are separated in two language models each of which carries the appropriate label "Danish" or "English".

The files written by the converting programs are composed of the second argument and two extensions. The first extension denote the type of grammar network: rtn, fsn, or wp (recursive transition network, finite state full grammar, finite state word pair grammar). The second extension denote the grammar format: sdl, voc, or icm (HTK SDL (StandarD Lattice), Vocalis Grammar Network format, and ICM format, for details we refer to the next section), e.g. mmuiALL.rtn.sdl (HTK SDL (StandarD Lattices) consisting of one main lattice and a number of sublattices), mmuiALL.fsn.sdl (weakly or approximately equivalent HTK standard lattice with expanded sublattices), mmuiALL.wp.sdl (approximately equivalent HTK SDL). In addition, the converter generates a file with only one extension .dic which is the HTK Dictionary format, e.g. mmuiALL.dic.

7.2.2 Recognition grammar formats

In general, when generating recognition grammars from unification based grammars (APSGs) the grammar converter goes through three steps:

(1) In a first step, feature sets are converted into labels denoting words (terminals) or labels referring to the head of other rules (non terminals). As APSG format described in Section 7.1.3 basically is a context-free grammar format (as defined within the Chomsky Hierarchy of Grammars), it is always possible to rewrite it as a weakly equivalent label based recursive transition networks (details are explained in Section 7.2.3). This first step leads to the internal rtn format which is written to the *.rtn.sdl, *.rtn.voc, *.rtn.icm files.

(2) In a second step, eventual recursions of the rtn format are removed and non-terminal are expanded. It is well-known that simple left-recursion and right-recursion of context free grammars can be rewritten as iterations in regular (finite state) grammars and that restrictions are limited to mid-recursions describing $a^n b^n$ sequences, "ab", "aabb", "aaabbb", etc. (eg. Chomsky 1956, p. 22, Chomsky 1957, p. 30). The converter rewrites eventual mid-recursions as two loops connected by a jump transition ("empty transition", "skip") which of course leads to the acceptance of a larger set of sentences ("abb", "aaabb", etc.). This second step leads to the internal fsn format which is written to the *.fsn.sdl, *.fsn.voc, *.fsn.icm files. Unless the rtn grammar involves mid recursion, fsn grammars are always weakly equivalent with the rtn grammar and, of course, with the input grammar. Otherwise, fsn grammar are only approximately equivalent (accepting a larger set of sentences).

(3) Finally, in a third step, a very compact wp grammar is build from the fsn format. A wp grammar is a finite state network (directed graph) where each state is associated with a word and outgoing transitions with the possible successors of this word. Each outgoing transition leads to the state associated with the word labelling the transition. Words

with the same set of successors can be collapsed to one state. The syntactic information submitted from fsn to wp grammars is restricted by the two word "windowing function". Consequently, wp grammars are only approximately equivalent with fsn grammars (unless, of course, the longest sentence accepted by the fsn consists of two words).

Each of the three internal grammar formats, rtn, fsn, and wp, are written to files of different formats. Currently, the grammar converter implements three writing procedures for recognition grammars. The recognition grammar formats are: (1) sdl: the HTK SDL (StandarD Lattice) format (Young et al. 1996, p. 291 ff.), (2) voc: the Vocalis speech recogniser format (currently not documented), and (3) icm: the ICM grammar network format (Bækgaard et al. 1992, p. 9 ff.) which is used by the CPK speech recogniser SUNCAR (Lindberg and Kristiansen 1995, p. 15 ff.). Writing procedures for other recognition grammar formats can easily be implemented.

In general, it seems not advisable to use the *.rtn.sdl, *.rtn.voc, *.rtn.icm grammars directly for recognition. The HTK SDL (StandarD Lattice) format supports nonterminals, however as "sub lattices" must be defined before they are referred to by nonterminals in other lattices, sdls cannot involve recursions (the Grammar Converter will only generate *.rtn.sdl. networks if there are no recursions). Viterby based speech recognition using "direct context free grammar constraint" (cf. Young et al. 1989, p. 11 ff.) is a bit misleading - unless we find it reasonable to talk about "recursive transition networks with no recursions". The Vocalis recogniser is claimed to support recursions, however currently it is not known if recursions are expanded, rewritten as iterations, or even a recognition algorithm superior to the standard Viterby Token Passing scheme has been worked out. So far, the *.rtn.voc format has not been tested. The SUNSTAR recogniser expands recursions and can principally read native ICM grammar files directly. However, *.rtn.icm files generated by the grammar converter differs from native ICM files as they are filtered for semantic features and lexical ambiguities and consequently are slightly more compact.

Basically, the method of expanding recursions into a predefined finite depth (as opposed to rewriting recursions as iterations) is not efficient. Expansions increase the size of the of the recognition grammars and the computational load of the Token Passing Scheme. Hence, the actual question is whether to use full grammars *.fsn.sdl, *.fsn.voc, *.fsn.icm or word pair grammars *.wp.sdl, *.wp.voc, *.wp.icm. Full grammars can in terms of number of states and arcs be huge in size, however as they (normally) submit all syntactic information from the external grammar to the language model, they are more efficient at constraining the search space. The so-called "perplexity" of full grammars is low. With respect to perplexity, see Brøndsted and Larsen (1994, p. 38) or Lee (1988, p. 132). Word pair grammars are very compact structures. However, perplexity is higher, as only a part of the syntactic information of the external grammar is submitted to the language model. In rough terms, the choice between full grammar or word pair grammar is a choice between "slow and good" or "fast and bad" recognition. When deriving fsn and wp language models, the Grammar Converter also computes details about the size ("slow-fast") and perplexity ("bad-good") of the networks. Both fsn and wp language models may involve iterations. This means that speech recognisers based on obsolete recognition algorithms like Level-Building cannot be supported by the grammar converter.

In general, the language models generated by the converters are transition networks ("directed graphs") where arcs are words represented orthographically or phonetically. The SetLexKey function described in Section 7.1.3 determines the representation. As the result of a recognition is given by an "optimal" path through the grammar network, the SetLexKey functions also affects the strings returned by the speech recogniser (orthographic or phonetic symbols). The current version of the grammar converter generates dictionaries or transcription tables simply by duplicating the word representations, e.g. the word "p ao l sp" consist of the phonetic symbols "p ao l sp", the word "paul" consists of "paul" etc. This means, that these tables can only be used if the lexical key is set to the feature denoting the phonetic transcriptions. Otherwise, the transcription tables must be generated by hand.

However, there are more serious reasons for using phonetic transcriptions as lexical key. We may consider the simple transcription table containing normal spelling and pronunciation variations of the Danish first name "Paul":

```
"paul"  p ao l sp, p oh l sp
"povl"  p ao l sp, p oh l sp
"poul"  p ao l sp, p oh l sp
```

Provided that the three lexical entries are syntactically equivalent (substitutable), a 1-best speech recogniser will only be able to recognise one of the entries and never the two others. In classic structuralism, text (orthography) and speech (phonetics) are considered to independent manifestations of language. In speech recognition and understanding technology, orthographic representations are actually superfluous and may as shown in the example even cause problems.

The language model used in the Campus Information System is a full grammar generated not by the grammar converter but "by hand" using grapHvite (see Chapter 6). This is due to the fact that the grammar was not ready for use as the Campus Information System was implemented. The choice of a full grammar is motivated mainly by the fact that the blackboard is currently not capable of processing semantic frame fragments. Semantic frame fragments may result from parsing a sentence accepted by an approximately equivalent word pair grammar but not fully accepted by the APSG grammar used for parsing. The use of a full grammar ensures that every sentence recognised by the speech recogniser leads to a complete semantic frame structure list generated by the parser. Consequently, the input-output-examples (4) and (9) in Section 7.1.2 cannot really occur in the Campus Information System in the current version.

7.2.3 Conversion strategy

As opposed to the working strategy of the parser (unification-based left-corner chart parsing, left-to right, bottom-up with top-down filtering) or the *typed text recogniser* (time synchronous Viterby-based pattern matching with token-passing), the working strategy of the grammar converter cannot be described with a single phrase and, perhaps, a few references. The problem dealt with in the converter seems to be "no man's land" between Computational Linguistics, Computer Science, Speech Technology, etc. and is consequently, to

our knowledge, rarely discussed in literature. In other speech understanding systems, language models seem to be produced independently of the grammar used for NLP. This section briefly and informally describes the steps leading from a unification-based APSG to a label-based RTN grammar. We consider the subsequent steps leading from the RTNs to the finite state full grammars and word pair grammars too trivial to describe further.

Unification grammars where features take values drawn from a finite set are basically context-free grammars as defined within the Chomsky Hierarchy of Grammars (Chomsky 1956). The APSG format described in Section 7.1.3 belongs to this class of unification grammars. This means that a strongly equivalent label-based context free grammar can be derived simply by iterative unification of "everything with everything" and subsequently identifying each unique feature set with a label. This approach, of course, do not work in real life, as even small "toy grammars" describing dates, months, years, hours, minutes, seconds, milliseconds, etc. may percolate values bottom-up and cause millions of feature sets to be created. This kind of percolation is presupposed by APSG grammars designed for generation of semantic frames without mapping. Consequently, the converter can only aim at weak equivalence when deriving the intermediate RTN format. Variables having no constraining function can be determined through iterative scanning of the entire set of axioms and b-rules and gradually removed. The remaining variables do either occur twice or more in the body of a rule (checking agreement) or can percolate values from the body to the head where they have the possibility of instantiating other variables occurring twice or more in a rule body (checking agreement). For each of these remaining variables, the converter computes a "variable declaration" enumerating all possible instantiations. The term "variable declaration" is a loan from the concept "feature declaration" of the EUROTRA Framework (see Section 7.1.3). The declaration tables enable the converter to calculate the size of arrays etc. needed for the conversion before actually allocating them. With a predefined threshold, the Grammar Converter can give up its venture, before trying to carry it out. Or it can resort to other strategies currently not implemented.

The basic idea of the working algorithm is that each rule can be expanded in a number of instantiations predicted by the (pruned) local variable declaration. After this step, further expansion can be based on simple matching, a Boolean counterpoint to more expensive unification in terms of memory and computational load. The result is an RTN where lexical ambiguities have been removed. For instance, parsing the sentence "show me pauls office" with the internally generated RTN gives the parse tree:

```
SYNTACTIC TREE NO 1
"axiom1" line 36 [
    "s1_imp" line 195 (s){  }[
        "vp_1" line 218 (vp){  }[
            "-" line 0 (show){  }
            "np_gen" line 238 (np_7){  }[
                "-" line 0 (me){  }
            ]
            "np_gen" line 238 (np_18){  }[
```

```
                "np_gen" line 238 (np_12){  }[
                    "-" line 0 (pauls){  }
                ]
                "np_gen" line 238 (np_18){  }[
                    "-" line 0 (office){  }
                ]
            ]
        ]
    ]
]
```

where labels occur in parentheses '()' facing empty feature sets enclosed in '{}'. Parsing the same sentence with the corresponding APSG (Appendix F), results in two parse trees:

```
SYNTACTIC TREE NO 1
"axiom1" line 36 [
    "s1_imp" line 195 (s){stype=imp }[
        "vp_1" line 218 (vp){vform=imp }[
            "-" line 165 (verb){lex=show,phon=sh ow sp,nb=$N,
                                valence=np_np, vform=imp,prs=$P }
            "np_pron" line 244 (np){case=acc,semtype=person,nb=sing,prs=P }[
                "-" line 175 (pron){case=$C=acc,lex=me,phon=m iy sp,
                                nb=$N=sing,prs=$P=1,subtype=person }
            ]
            "np_gen" line 238 (np){case=no,semtype=place }[
                "np_person1" line 248 (np){case=gen,semtype=person,id=pd }[
                    "-" line 72 (proper){case=$C=gen,semtype=person,
                  surname=no,lex=pauls,phon=p ao l z sp|p oh l z sp,id=$I=pd }
                ]
                "np_place2" line 261 (np){case=$C=no,semtype=$S=place,
                                        id=office,nb=$N,prs=$P,def=no }[
                    "-" line 154 (noun){case=$C=no,semtype=place,lex=office,
                                phon=oh f ih s sp,id=$I=office,nb=sing }
                ]
            ]
        ]
    ]
]
SYNTACTIC TREE NO 2
"axiom1" line 36 [
    "s1_imp" line 195 (s){stype=imp }[
        "vp_1" line 218 (vp){vform=imp }[
            "-" line 165 (verb){lex=show,phon=sh ow sp,nb=$N,
```

```
                                        valence=np_np, vform=imp,prs=$P }
            "np_pron" line 244 (np){case=acc,semtype=person,nb=sing,prs=P }[
                "-" line 175 (pron){case=$C=acc,lex=me,phon=m iy sp,nb=$N=sing,
                                                prs=$P=1,subtype=person }
            ]
            "np_gen" line 238 (np){case=no,semtype=place }[
                "np_person1" line 248 (np){case=gen,semtype=person,id=pmck }[
                    "-" line 112 (proper){case=$C=gen,semtype=person,
                surname=no,lex=pauls,phon=p ao l z sp|p oh l z sp,id=$I=pmck }
                ]
                "np_place2" line 261 (np){case=$C=no,semtype=$S=place,
                                            id=office,nb=$N,prs=$P,def=no }[
                    "-" line 154 (noun){case=$C=no,semtype=place,lex=office,
                                    phon=oh f ih s sp,id=$I=office,nb=sing }
                ]
            ]
        ]
    ]
]
```

denoting the ambiguous nature of the sentence (which "Paul"?).

Historically, the introduction of unification grammars in the eighties (Kay 1985, Shieber 1986) took the sting out of Chomsky's main objection against the application of context free grammars in natural language descriptions, namely the objection that such grammars would require a huge, and possibly infinite, number of symbols (Chomsky 1957). The grammar converter actually calculates and creates these symbols, here denoted "labels", which would correspond to the externally defined unification grammar. This, of course, leads to the question when and under which circumstances the grammar converter exceeds the threshold where conversion must be considered "unrealistic". In general, we think that the most important parameter is the number of "syntactic word categories" defined in the unification grammar. By "syntactic word category" we mean a group of words each of which can substitute any other member in the same category in all syntactic contexts. In the Campus Information System grammar in Appendix F, words like "speech_coding_lab", "speech_lab", "dialogue_laboratory", etc. are syntactically equivalent and constitute one syntactic category though they have different "meanings". By way of contrast, "brondsteds" conforms to an own syntactic category. In a context like "show me brondsteds office" it can be substituted by many other words ("dalsgaards", "mckevitts", etc.), however in other contexts, e.g. "show me tom brondsteds office", it cannot be substituted by any other word.

7.3 Typed text recogniser

The *typed text recogniser* is a device which simulates a speech recogniser. It is intended for fast evaluation of subgrammars during the design phase. In terms of input-output, this recogniser takes a typed sentence or phrase as input and returns a sentence/phrase which is at the same time "similar" to the input and grammatical according to a predefined subgrammar. The recognised sentence is passed on to the parser (see Section 7.1). The subgrammar used by the typed text recogniser is a finite state transition network automatically generated from the compound feature based subgrammar used by the parser. The finite state transition network is equivalent to the "(sub)language model" used by the speech recogniser of the run-time system. The transition networks may be generated using the grammar converter described in Section 7.2.

The algorithm of the text recogniser is based on exactly the same Viterby based optimal search as a modern standard speech recogniser. It processes one character (equivalent to a "time frame" in a speech signal) at a time and measures the "distance" between this character and each character of each word in the lexicon. Hence, the textual representations of words function like Markov Models in a speech recogniser. To simulate garbage modelling, pseudo words ('*'s, garbage characters) are inserted into the finite state transition network. The distance between '*' and any other character is set to a value between the distance between two identical characters (0) and the distance between two different characters (the arbitrary value 100). The distance calculation corresponds to probability density functions in Markov-based speech recognition or Euclid in dynamic time warping (DTW) (Young et al. 1989). Local distances are accumulated to global distances and stored in so-called tokens which are passed through the finite state grammar network ("token-passing").

A simple input/output example using a finite state grammar derived from the APSG of Appendix F is shown below:

```
1: pointtothelibrary
2: grammar English, score -49.000000
3: [0 point 4][5 to 6][7 the 9][10 laboratory 16]
4: point to the laboratory
```

1 contains the typed input (spaces, if any, are ignored by the decoding algorithm), 2 shows which subgrammar submits the best match to the input and with which score (this score can be used for a rejection threshold), 3 shows the segmentation of the input string by the decoding algorithm (character 0-4 matches `point`, 5-6 matches `to`, etc.), and 4 the actual recognised string. As `library` is not within the defined vocabulary, the recogniser cannot match this word properly. The text recogniser was used in the design phase of the Campus Information System to simulate "ideal" speech recognition, where input covered by the application subgrammar is always recognised correctly.

7.4 Random sentence generator

The *random sentence generator* is intended for fast evaluation of subgrammar covering. The program generates a set of random sentences which are accepted (or covered by) the subgrammar being evaluated. The generator works on the internal RTN format generated by the grammar converter (see Section 7.2).

7.5 Natural language generation

A *natural language generator* which takes semantic frames as input and outputs a sentence to be passed on to a speech synthesiser is under development. The generator will use the same grammar definitions as the parser and can in terms of input-output be considered the reverse counterpart of the parser. In the current prototype of CHAMELEON little emphasis have been put on natural language generation. The generation is based on simple rules, generating static canned answers to the different types of questions that can be posed to the system.

7.6 Summary

Here, we have described the natural language processing (NLP) module of CHAMELEON. The central components of the NLP module are (1) natural language parser, (2) grammar converter and (3) typed text recogniser. The parser takes output from a speech recogniser and returns semantic frame representations. The converter converts the grammar used by the parser into a language model to constrain the search space of the speech recogniser. The typed text recogniser simulates speech recognition. Little work has been done on natural language generation, and at present CHAMELEON generates canned responses, so generation will be a focus of future work.

Chapter 8

Topsy

Natural language Processing (NLP) is one of many examples, where traditional Artificial Intelligence (AI) has not yet been able to deliver a convincing demonstration of intelligence. There are in fact quite a few powerful programs that process natural language, but each of these solutions are "partial". The fact that one must explicitly come up with rules, extend grammars and so on, is a huge impediment in creating truly intelligent systems. Such a system will never show more intelligence than what oneself puts into it by hand. In the end, one ends up having to pre-specify the situations that the system in question is going to handle, extended perhaps to rules for creating new "situation rules", according to pre-specified meta-rules and so on. Therefore, stepping away from such tedious work and partial grammars will, in our view, improve the intelligence shown by a system in every respect. In the (sub)system we describe in this section, the only rule of behavior is to represent co-occurrence vs. exclusion of sensorial input. Hereafter, pure-process concurrency and a new and unique kind of hierarchy supply associative processing of sensorial impressions automatically. This system's learning is also based on this same universal rule. We thus avoid pre-specification of everything but the overall pattern for the organization of experience.

NLP is still one of the most difficult areas in the field of AI. Many of the AI approaches to NLP use a grammar to parse sentences into tree-like data structures such as that discussed in the previous chapter. These structures are then augmented by various types of semantic processing. Pragmatics is also sometimes taken into consideration. Whether the traditional AI approaches to NLP treat syntax, semantics and pragmatics to an equal degree or not, they all have one thing in common: the AI function dictates, a priori, using explicit grammars, the global behavior of the system. Among others, an important drawback that follows using grammars is that the AI function incorporated into such systems is used towards generating intelligent behavior that actually bears no relationship to the methods by which intelligence is achieved in natural organic systems. Were these systems wildly successful, one could perhaps ignore this difference. However, being adaptive, anticipating and learning from own experience eliminates the need of specifying explicit grammars in NLP.

8.1 Doing NLP with Topsy

The Phase Web (Manthey 1998a,b) is currently implemented by the Topsy system - a general purpose, distributed, goal-oriented program that represents knowledge as patterns of synchronization. A Phase Web system builds up its knowledge structure by employing a universal rule that registers co-occurrence and exclusionary relationships among its sensors. The current implementation of Topsy includes the following basic elements:

Sensors: A sensor in Topsy represents a sensor situated in the real world, and can be in one of the two possible phases (states): plus (+) or minus (-), denoting respectively the presence or absence of a corresponding stimulus. We denote the opposite value of a sensor with an overbar: `Q` vs. $\overline{\texttt{Q}}$. Information flows strictly from the sensor(s) towards a Phase Web system.

In the NLP realm, sensors might represent the frequency bands, phonemes, words, 'me/other' (agent) distinctions, and the like. In CHAMELEON and the IntelliMedia WorkBench, they generally represent words or intentions, derived by the upstream real-time speech analysis and NLP modules and communicated to Topsy in the form of semantic frames.

Effectors: An effector allows a Phase Web system to interact with the real world. In order to change the phase of a sensor, an effector interacts with the environment and carries out an operation that will ultimately change a corresponding sensor's state, provided that there is a goal issued for changing the sensor in question. An effector may also induce side effects; i.e. when changing it's associated sensor (in the real world) the phases of other sensors could also be changed as a result hereof.

In CHAMELEON and the IntelliMedia WorkBench, an effector causes the generation of output frames to the downstream speech synthesis and laser-pointing modules.

Actions: An action is created whenever a co-exclusion (explained below) is found, and carries out transformations between the preconditions and postconditions (the action-defining co-occurrences) found by an Event Window[1]

The concept of a co-exclusion-based action in the Phase Web is based on complementary, and hence mutually-exclusive, sets of co-occurring sensor values. For example, the two co-occurrences $\overline{\texttt{Q}}$ `+ A and Q +` $\overline{\texttt{A}}$ are such a complementary pair of co-occurring sensor values. Their 'outer product' produces the action `QA`, for which one of the defining co-occurrences constitutes the action's pre-condition, and the other the corresponding post-condition. Which is which depends on the dynamic context.

Note particularly that `QA` itself possesses an orientation (+/-), ie. it is both an action *and* a *meta*-sensor. Therefore, actions in general can be co-excluded, opening the door to a hierarchy of increasingly refined distinctions and contexts.

[1] An Event Window, explained next, automatically (and efficiently) captures co-exclusions occurring dynamically in the input.

Event Windows : An event window defines the amount of time necessary/convenient to perceive the change of phase of two or more sensors as an atomic event. When this occurs, the corresponding action is created. There may in principle be multiple Event Windows active simultaneously.

The logic behind the Event Window mechanism is that if two (or more) sensor phase *changes* are simultaneously present in the window, then their current respective values must necessarily be complementary to the values they had before they entered the window, thus fulfilling the definition of an *action* above. Event Windows are Topsy's basic learning mechanism, and are very efficient. The reader finding the above summary opaque is referred to Manthey (1991, 1994, 1997a,b, 1998a,b,c) at this point.

8.1.1 Question/answer example

We now present a simple example in order to show how the above-mentioned Phase Web elements relate to each other. The world depicted is a conceptual one and consists of two natural-language constructs: a question **Q** and an answer **A**. Before anything else is said, assume that anyone able to get a sensorial impression from this world will sense either the presence of a question or the presence of an answer, as these natural-language constructs exclude each other's existence[2]. This simple world can be modeled in Topsy by two sensors: `Q` and `A` respectively. Figure 8.1 shows sensors `Q` and `A` and their phases (+ or -) together with the action/meta-sensor `QA` they form. An effector `Eff A` is attached to sensor `A` and may be used to deliver an answer to the world.

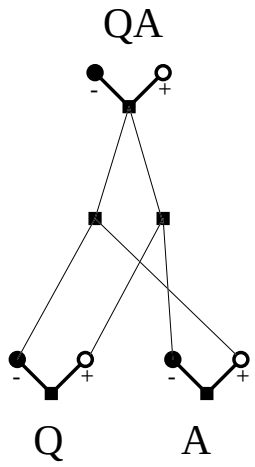

Figure 8.1: A Question-Answer world modeled in Topsy by two sensors. The filled circles denote the current state (here, all '-'), and the boxes in the middle the co-excluding co-occurrences.

[2]As said before, a sensor has two phases, thus one can envision the co-occurrence $\overline{\texttt{Q}}$ + $\overline{\texttt{A}}$. If this co-occurrence is possible so presumably is `Q + A`, thus the co-exclusion $(\overline{\texttt{Q}}$ + $\overline{\texttt{A}}) \leftrightarrow$(Q + A) can be constructed. As this co-exclusion is not relevant for our example, it will be disregarded.

First, an Event Window is defined, and watches for changes in the sensors Q and A. Then, when the sensors change their phases during the Δt period, the Event Window registers the changes by instantiating an action over sensors Q and A, yielding QA. During learning, Topsy builds the (trivial) hierarchy shown in Figure 8.1 and ends up with a meta-sensor QA representing an exclusionary relationship on top of the co-occurrences seen during the Δt period[3].

Assume now, after this learning, that the world changes and that Q is now present: there is a question to be answered (i.e. the world is in the state Q + $\overline{\text{A}}$). In order to deliver an answer, Topsy has two possibilities of behavior: either make an answer "appear" or make the question "disappear". At this stage, the phase of sensor Q is (+) while A's phase is (-). Let's suppose we choose the strategy of causing an answer to "appear" into the world. In order to do that, a goal must be issued on the sensor A, requesting A's change of phase. This implies that the effector Eff A becomes relevant. The action QA namely *defines* that changing the phase of A to (+) implies changing the phase of Q to (-). As a consequence, the action volunteers a goal to change Q's phase as well, which results in the activation of the effector Eff A. Ultimately, this effector delivers the answer (which here is unspecified) to the world, causing the current stimulation of Q to disappear as well.

8.2 Integrating Topsy into CHAMELEON

In order to integrate Topsy into the CHAMELEON platform, a number of aspects must be considered. The Campus Information System domain presently addresses, due to its infancy, a fairly limited set of NLP problems, although as pointed out in Chapters 2 and 9, the system is open to extension. The system's openness to change is ensured by the frame semantics. As soon as there is a need to extend the system, new modules can be integrated, as long as these are able to subscribe to the blackboard or other modules. As a consequence hereof, the new modules must make use of the frame semantics. Thus, the first aspect to consider when integrating a Topsy module into CHAMELEON is that of interfacing. The new module must understand the structure of the input frames and also be able to deliver similar output. The design of the Topsy module must also be open to change, as new aspects of NLP might be added in future versions. In the following sections, we present our approach to integrating Topsy into CHAMELEON.

8.2.1 Topsy's environment

Recall that a Phase Web compliant system senses the surrounding world, looking for (and learning) events that co-occur or exclude each other. Given our initial problem definition, the world (called OuterWorld) from which the Topsy module gets sensorial impressions is defined by the following entities:

[3]In this particular case (cf. Figure 8.1) the abstraction found is: (Q + $\overline{\text{A}}$)↔($\overline{\text{Q}}$ + A) and the current global state of the world ($\overline{\text{Q}}$ + $\overline{\text{A}}$) shows that QA is not "relevant".

- Input frames delivered to the blackboard from NLP or gesture/pointing modules. These frames define and model the semantics and intention of a user utterance or gesture.
- Output frames generating the system's intention and semantics in order to deliver output to the speech synthesizer and laser modules.
- Integration frames representing cumulative semantics and intentions constructed by modules processing frames found on the blackboard.

In the following sections, Topsy's internal representation of this OuterWorld will be called TopsyWorld.

If Topsy were to act as, for example, a dialogue manager it must display behavior expected of such a typical manager.

Three main steps characterize our approach with respect to solving the initial problem:

- Appropriate co-exclusions must be identified in the OuterWorld. The analysis must reveal candidates for Topsy sensors.
- Effectors are Topsy's means of interaction with the OuterWorld. A decision as to which effectors are needed must be taken.
- A suitable learning script must be devised.

The following sections discuss each of these aspects.

8.2.2 Filtering the OuterWorld

Topsy's sensors are, by definition, situated at the boundary between Topsy itself and the OuterWorld. In order to connect TopsyWorld to OuterWorld, an interface containing the chosen sensors must be constructed. Besides the sensors, this interface must also support communication with the blackboard module. As the OuterWorld consists of frames in its entirety, the place to look for co-exclusions is therefore the fields contained in these frames. Let us therefore take a closer look at an example user utterance and its associated input frame:

```
USER: Show me Paul Dalsgaard's office!

FRAME (produced by NLP module):

nlp(
   intention(instruction(pointing)),
                location(person(pd),
```

```
                    type(office)),
                          time(T)
)
```

As far as Topsy is concerned, this frame contains both useful and redundant information. More precisely, only the arguments of each field qualify as candidates for being a sensor. The field identifiers: `nlp`, `intention`, `type` and `time` appear in every input frame so there is no need for Topsy to sense for these identifiers, as sensors representing their occurrence would always have a positive phase. Therefore such field identifiers are filtered out from TopsyWorld. We thus strip unneeded information out at the interface level, as shown in Figure 8.2.

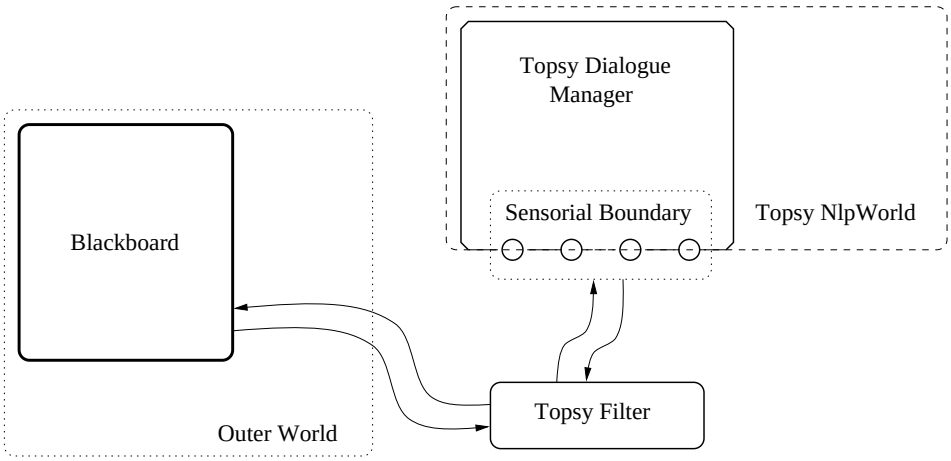

Figure 8.2: Filtered OuterWorld for Topsy

The relevance of the argument `T` from the `time()` field in the context of TopsyWorld is questionable. This value is typically used is for recognising synchronized input and producing synchronised output with respect to the various input and output modules. We therefore strip `T` from TopsyWorld, resolving it instead at the Topsy filter level.

8.2.3 Co-occurrences in OuterWorld

In this section we identify the critical co-exclusionary structures in OuterWorld. Presently, the Campus Information System's dialogue-repertoire is quite limited. Usually one will regard it as a traditional information browser system, with the objective of answering user's questions regarding employees and their office location. Now let us look at the system through "Phase Web eye-glasses". The main actors in the application are persons, locations and routes. These actors are all playing on the IntelliMedia WorkBench stage, but who is directing them, and based on which rules? In fact, there is no director! Rather, in the "play" as seen through our Phase Web eyeglasses, the actors just happen to co-occur with one another. Or exclude each other. Appropriate Event Windows automatically

pick up the actors' (sensorial) co-occurrences and implicit co-exclusions, and generate the corresponding hierarchical knowledge structure.

Notation

In the sections to come, we will use the following notation:

- `(PERSON)` and $(\overline{\texttt{PERSON}})$ will denote a sensor's phase as being `(+)` or `(-)` respectively. There will of course be many instances of this sensor - one each for each person, and similarly for locations, etc.
- The tuple `(PERSON, LOCATION)` represents the co-occurrence of sensors `(PERSON)` and `(LOCATION)`. The phases of both sensors here are `(+)`.
- The relation denoted as $\texttt{(QUESTION, }\overline{\texttt{ANSWER}}\texttt{)} \leftrightarrow \texttt{(}\overline{\texttt{QUESTION}}\texttt{, ANSWER)}$ will denote an (arity 2) exclusionary relationship between sensors `(QUESTION)` and `(ANSWER)` as depicted in Figure 8.1.
- Similarly, the relation

  $\texttt{(QUESTION, PERSON, }\overline{\texttt{ANSWER}}\texttt{, }\overline{\texttt{LOCATION}}\texttt{)} \leftrightarrow$
  $\texttt{(}\overline{\texttt{QUESTION}}\texttt{, }\overline{\texttt{PERSON}}\texttt{, ANSWER, LOCATION)}$

  represents an arity-4 co-exclusion.

When looking at the input examples given in Chapter 2, one sees that the questions issued to the system can be categorized as:

1. **Intentions about employees' offices**

   This category is the class of reciprocal associations between employees and their respective offices. Speaking in Phase Web terms, the relationship between employees and offices is a clearly defined co-occurrence relationship. In fact, we have here the central co-occurrence relationship in the Campus Information System domain, namely: `(EMPLOYEE, OFFICE)`.

2. **Intentions spiced with deictic expressions**

   This category is the classic NLP problem of deixis. The question "Who's office is this?" is a hard nut to crack, as there is no indication as to which office "this" refers to. In the Campus Information System, the Dialogue Manager module consults the blackboard for the presence of input frames from the gesture module. If no additional information is found on the blackboard the dialogue manager module can ask the user for help.

   In the current version of CHAMELEON Topsy does not receive vision input and so must always ask the user in this case.

If Topsy were so connected, however, it could resolve "this" by setting the sensor DEICTIC_THIS to (+). The appropriate co-occurrences are then (QUESTION, DEICTIC_THIS, ANY_SENSOR). Notice how the integration of the information purveyed by multiple media occurs painlessly via the co-occurrence relation.

3. **Intentions about routes**

   This category is the association between two places (termed the origin and the endpoint of a route) and a number of points (representing the path). This relationship between points is another co-occurrence relationship in the Campus Information System, namely:

   (ORIGIN, POINT, POINT, ..., POINT, ENDPOINT).

8.3 Our solution

In the previous section we identified several co-occurrence classes that will serve as our point of departure for building Topsy into CHAMELEON. The relevant exclusionary relationships are now due for investigation. The little Question-Answer world, introduced earlier, will be extended in the following.

We have seen that the presence of a question excludes the presence of an answer, but this is too general and must to be tailored to our needs. In order to do this, each person (e.g. Paul Dalsgaard), office (e.g. A2-202), and intention type (e.g. Query) is modeled in TopsyWorld by a sensor indicating its presence. The fact that Paul has office A2-202 is expressed as a co-exclusion between the following co-occurrences:

$$(\mathtt{PAUL},\ \mathtt{A2\text{-}202}) \leftrightarrow (\overline{\mathtt{PAUL}},\ \overline{\mathtt{A2\text{-}202}})$$

which in turn, combined with the exclusion shown in Question-Answer world, concludes that Topsy must model the following co-exclusion when expected to answer questions about employees' offices:

$$(\mathtt{QUESTION},\ \mathtt{PAUL},\ \overline{\mathtt{ANSWER}},\ \overline{\mathtt{A2\text{-}202}}) \leftrightarrow$$
$$(\overline{\mathtt{QUESTION}},\ \overline{\mathtt{PAUL}},\ \mathtt{ANSWER},\ \mathtt{A2\text{-}202})$$

In order to achieve deictic resolution, the following co-exclusion is relevant:

$$(\mathtt{QUESTION},\ \overline{\mathtt{ANSWER}},\ \mathtt{DEICTIC\_THIS},\ \overline{\mathtt{ASK\_USER}}) \leftrightarrow$$
$$(\overline{\mathtt{QUESTION}},\ \mathtt{ANSWER},\ \overline{\mathtt{DEICTIC\_THIS}},\ \mathtt{ASK\_USER})$$

As pointed out above, when a deictic question is encountered, Topsy currently asks the user for assistance. As soon as the user specifies the office he is referring to, Topsy will be able to deliver the answer. We note that this scenario corresponds in the end to a user asking the system: "*Who* lives in office LOCATION?". Answering requests about an

office's tenant is a finger snap for Topsy, as explained in the following. The structure that Topsy builds is a web of intertwined exclusionary and co-occurrence relationships, so the knowledge structure it builds is a rather subtle one. If Topsy has once seen the co-exclusion (PAUL, A2202)↔($\overline{\texttt{PAUL}}$, $\overline{\texttt{A2202}}$), it will be able to deduce answers to questions regarding both the tenant and the office. So, answering "Who lives in office LOCATION?" is, seen from Topsy's standpoint to be the same as answering "Where lives Paul?", due to the fact that both questions address entities belonging to the same co-exclusion. Another way to view this is that Topsy's knowledge structure implicitly yields associative look-up.

In order to answer a user's questions regarding a route, the following co-exclusion is needed:

(ROUTE, ORIGINx, ENDPOINTx, $\overline{\texttt{ROUTEx}}$)↔($\overline{\texttt{ROUTE}}$, $\overline{\texttt{ORIGINx}}$, $\overline{\texttt{ENDPOINTx}}$, ROUTEx).

where x designates a specific office or route.

8.3.1 Which sensors need effectors?

Assume that a Topsy module is constructed based on the co-exclusions found in the previous section. In order to interact with TopsyWorld, the module must be equipped with effectors. In this section we will identify the effectors needed by Topsy.

Let's start by looking at Figure 8.3a. The figure shows the hierarchy built by Topsy while trained to answer the question "Show me Paul's office!".

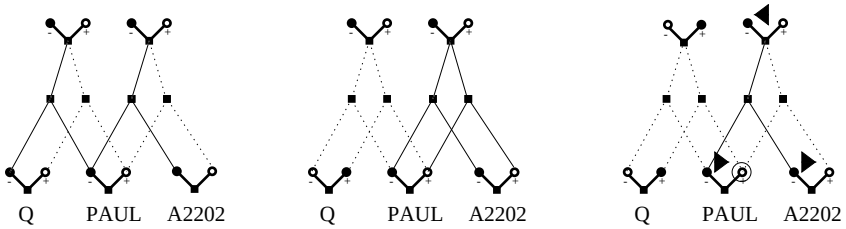

Figure 8.3: Knowledge hierarchy build by the Topsy module when trained to answer the question "Show me Paul's office!".

The sensors are in place, set at their (-) phase, the knowledge hierarchy is woven and Topsy is ready to accept queries about Paul or his office. As soon as a user issues a query modeled by Topsy, sensor Q representing the the presence of a question in TopsyWorld will be set at its (+) phase as shown in Figure 8.3b. Now an *impulse*[4] is (automatically) set on

[4] An *impulse* is an externally supplied (here, by the Filter interface) sensor-level goal that is then bubbled-up the knowledge hierarchy until it reaches the most general action that can accomplish the original sensor-level goal.

the sensor (PAUL) as an indication to Topsy that we wish to change the phase of (PAUL).

Changing the phase of (PAUL) implies that the sensor(s) seen to co-occur together with (PAUL) in various co-exclusive relationships, will also change their phase to (+). The impulse bubbles up to the highest relevant place in the hierarchy, where an action then takes a decision about changing the phase of (PAUL), as shown in Figure 8.3c. If there is more then one meta-sensor that is able to take a decision, one of these is chosen (currently) at random. Depending on the decision taken, a goal (the black arrowhead) is issued and distributed downwards in the hierarchy. In the end, a number of base-level sensors, in fact all sensors pertinent to a co-exclusion that has (PAUL) in it, will receive the goal of changing their phase, as shown by the arrowheads in Figure 8.3c. At this point the sensors marked for change will fire their effectors, if there are any. We can see that only information regarding employees and offices is relevant for us, therefore, only these sensors will be equipped with effectors.

8.3.2 Training

Topsy must be trained in order to build its knowledge hierarchy. We envision the training scenario depicted in Figure 8.4. Topsy is connected in parallel with an alternative Dialogue Manager module in order to "tap" the ongoing communication. Both the input and the output frames that flow between the blackboard and the Dialogue Manager are relevant for Topsy, as they contain relevant co-occurrences. An input frame is, as a general rule, always followed by a corresponding output frame. Thus Topsy will easily model this aspect as a co-exclusion. Figure 8.4 shows how unneeded information from the input or output frames is filtered out and then the actual co-occurring sensor changes so derived are presented to Topsy.

Topsy will react as follows: If Topsy receives new information (a new co-occurrence), the knowledge hierarchy will be augmented accordingly via the Event Window $\rightsquigarrow$ action $\rightsquigarrow$ meta-sensor process. If, on the other hand, information that has already been seen arrives, Topsy has two possibilities of behaviour:

(1) A plain ignoring of the newly arrived information, as Topsy ensures the "once-is-enough" principle: any co-occurrence (actually, co-exclusion) is implicitly "memorized".

(2) Depending on the number of knowledge hierarchy levels that one wishes Topsy to build:

The Base Level only: If only a Base Level is to be constructed, then memorizing again an already seen co-occurrence would not do any good, as this information is redundant.

The Base Level and N upper levels ($N \geq 1$): Assume for a moment that we've trained a Topsy module, and the result of the training is the knowledge hierarchy shown in Figure 8.3a, and that we have instructed Topsy to build a knowledge hierarchy with two levels (a Base Level and one upper level). Showing the same information again will in this case create a meta-sensor on Level 1 as shown in Figure 8.5.

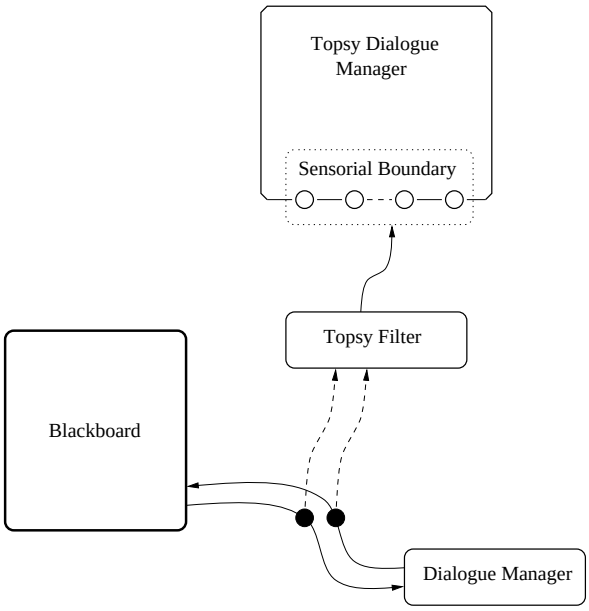

Figure 8.4: Training a Topsy module: The module is "tapping" the ongoing communication in the system. As soon as a relevant frame is detected by the Filter, the frame is stripped, and the gist presented to Topsy.

In fact, for each time the same information is presented to Topsy, a new meta-sensor is (can be) created on the next upper level. The knowledge hierarchy's information content remains the same, but now the information repeatedly shown to Topsy is becoming increasingly refined, as explained next.

Assume a Topsy hierarchy built only at the Base Level. Now, if a sensor has been seen to take part in two different co-occurrences that do not exclude each other, say `(PAUL, A2-202)` and `(PAUL, PROFESSOR)`, then each co-occurrence will be modeled in the knowledge hierarchy by distinct meta-sensors situated at the next level, cf. Figure 8.5. Assume that an impulse is set on `(PAUL)` and recall that the impulse bubbles up until a meta-sensor can take a decision. Now that we have two meta-sensors that are able to take the decision, only one of them will be chosen (currently, randomly) by the system. Without a given context, this would not create problems because both facts about Paul are true: he lives in A2-202 and he is a professor. If we are in a given context, say we are asking about Paul's office, then we would thought be greatly surprised to get information about his job! So, in order to model context and ensure a better answer then a randomly chosen fact, we have to go further up the hierarchy. This can be done by repeatedly presenting the same information to Topsy while training. In the case at hand, there would presumably be other similar and relevant meta-sensors which have been co-excluded with those shown in Figure 8.5, which co-exclusions would establish the context for answering the question.

Topsy was trained by presenting it with scripts containing information, as would have been generated by the Campus Information System. We have generated the frames as

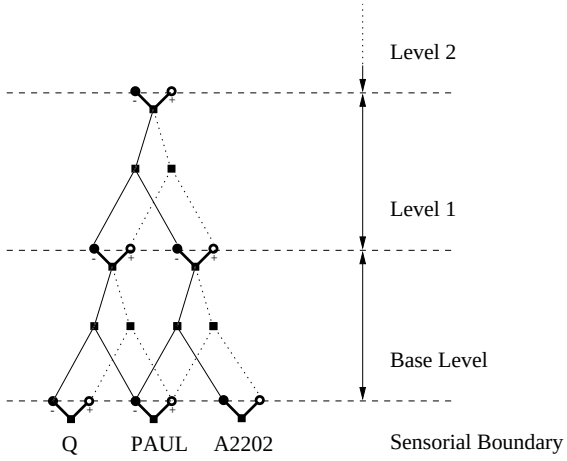

Figure 8.5: When training a Topsy module, repeatedly showing the same information will extend the knowledge hierarchy: for each time the same information is presented to Topsy, a new meta-sensor is (can be) created on the next upper level.

they would be found on CHAMELEON's blackboard. The frames were filtered and question/answer pairs, e.g. `(QUESTION, PAUL)` followed by `(PAUL, A2-202)` were presented to Topsy. One presentation sufficed to produce the hierarchy in Figure 8.3a. If thereafter Topsy were presented with `(QUESTION, PAUL)` it would respond with `(PAUL, A2-202)`. The same answer would be given to `(QUESTION, A2-202)`, via the common association of `(QUESTION)` and `(A2-202)` with `(PAUL)`.

8.4 Summary

We have described how the Phase Web paradigm, as embodied in the Topsy system, processes example utterances from the Campus Information System domain on the basis of its own experience. By relying on learning, not only is the system *self*-extending, but we also avoid having to put in myriads of linguistic and gestural 'facts' by hand.

The knowledge structure needed to represent and respond to these queries is quite trivial, and indeed could be replaced by a simple, traditional database module or some Pascal programs. However, we hope we have succeeded in communicating the generality and power of the learning paradigm, such that it is apparent that more difficult NLP problems, requiring processing that is beyond database lookups, can also be handled. We hope of course to pursue this promise.

One example of this promise that even the present very simple knowledge structure can answer - quite unintentionally - "Who lives in `LOCATION`". This reflects the emergent properties implicit in the Phase Web's unique hierarchical representation. In the same spirit, we note that the co-exclusion relationship also captures 'negative information' - what's *not* the case - that is unavailable to much of traditional NLP (von Hahn 1997).

Finally, we noted very briefly how the problem of integrating information from multiple

media is solved trivially and directly by registering their co-occurrence.

Chapter 9

Conclusion

We started by introducing IntelliMedia 2000+ and its requirements for an IntelliMedia software demonstrator to be used for both research and education purposes. We showed how a number of candidate applications for a general *CHAMELEON* software platform to integrate the processing of spoken dialogue and images were discussed, resulting in an *IntelliMedia WorkBench*. We described the architecture and implementation of CHAMELEON: an open, distributed architecture with ten modules glued into a single platform using the DACS communications system. Some of the modules are commercial products (speech recogniser/synthesiser), others are customised products (e.g. laser), some are our existing software customised (e.g. NLP module, Topsy), and we have developed some from scratch (e.g. gesture recogniser, domain model).

We described the IntelliMedia WorkBench application, a software and physical platform where a user can ask for information about things on a physical table. The current domain is a *Campus Information System* where 2D building plans are placed on the table and the system provides information about tenants, rooms and routes and can answer questions like "Whose office is this?" in real time. We have described the frame semantics, in both abstract and implemented terms, which is the meaning representation of input and output information constructed on CHAMELEON's blackboard during the course of a dialogue interaction. A sample dialogue currently processed by the IntelliMedia WorkBench was shown and a look at the blackboard frame semantics in both theory and practice. Also, we have described in detail how the various modules perform their tasks.

Turning to the requirements of a demonstrator (Chapter 1, Section 1.4.1) it is noted that CHAMELEON fulfills all five requirements. We have shown that CHAMELEON fulfills the goal of developing a general platform for integration of at least language/vision processing which can be used for research but also for student projects as part of the Master's degree education. Also, the goal of integrating research from four research groups within three Departments at the Institute for Electronic Systems has been achieved.

9.1 Relation to other work

IntelliMedia is an area which has had an upsurge of new interest, particularly since 1994 (Mc Kevitt 1994, 1995/1996). Of particular relevance to our work here is other work on developing IntelliMedia platforms and in particular: *Situated Artificial Communicators* (Rickheit and Wachsmuth 1996), *Communicative Humanoids* like *Gandalf* (Thórisson 1996, 1997), *AESOPWORLD* (Okada 1996, 1997) and MultiModal Interfaces like *INTERACT* (Waibel et al. 1996).

Situated Artificial Communicators (SFB-360) is a collaborative research project at the University of Bielefeld, Germany which focusses on modelling that which a person performs when with a partner he cooperatively solves a simple assembly task in a given situation. This involves acoustic perception of the spoken word, visual perception of the partner and the objects and processes involved in the situation, understanding of that perceived, formulation of own utterances (e.g. instructions) and the planning of actions. The domain of application is one where two communicators cooperatively construct an object. The instructor has a diagram and instructs the other, the constructor, to carry out actions. The object chosen is a model airplane (Baufix) to be constructed by a robot from the components of a wooden building kit with instructions from a human. SFB-360 comprises ten subprojects in four thematic fields: (1) speech and visual perception, (2) perception and reference, (3) knowledge and inference and (4) speech-action systems.

SFB-360 is a well rounded project considering all the issues involved in developing a MultiModal platform. SFB-360 includes equivalents of the modules in CHAMELEON although there is no learning module competitor to Topsy. What SFB-360 gains in size it may loose in integration, i.e. it is not clear yet that all the technology from the subprojects has been fitted together and in particular what exactly the semantic representations passed between the modules are. The DACS communications system which we met in Chapters 2 and 3 and currently use in CHAMELEON is a useful product from SFB-360.

Gandalf is a communicative humanoid which interacts with users in MultiModal dialogue through using and interpreting gestures, facial expressions, body language and spoken dialogue (Thórinson 1996, 1997). Gandalf is an application of an architecture called *Ymir* which includes perceptual integration of multimodal events, distributed planning and decision making, layered input analysis and motor-control with human-like characteristics and an inherent knowledge of time. Ymir is a broad model of psychosocial dialogue skills that bridges between multimodal action and multimodal perception in a coherent framework. It is a distributed modular approach to perception, decision and action and can be used to create autonomous characters like Gandalf capable of full-duplex multimodal perception and action generation. Ymir has a blackboard architecture and includes modules equivalent to those in CHAMELEON. However, there is no vision/image processing module since gesture tracking is done with the use of a data glove and body tracking suit and an eye tracker is used for detecting the user's eye gaze. Also, Ymir has no learning module equivalent to Topsy. Ymir's architecture is even more distributed than CHAMELEON's with many more modules interacting with each other. Also, Ymir's semantic representation is much more distributed with smaller chunks of information than our frames being passed

between modules.

AESOPWORLD is an integrated comprehension and generation system for integration of vision, language and motion (Okada 1996, 1997). It includes a model of mind consisting of nine domains according to the contents of mental activities and five levels along the process of concept formation. The nine domains are (1) sensor, (2) recognition-understanding, (3) planning-creation, (4) action-expression, (5) actuator, (6) desire-instinct, (7) emotion-character, (8) memory-learning, and (9) language and the five levels are: (1) raw data, (2) cognitive features, (3) conceptual features, (4) simple concepts, (5) interconnected-synthesized concepts. Vision and motion are connected and controlled by planning. The system simulates the protagonist or fox of an AESOP fable, "the Fox and the Grapes", and his mental and physical behaviour are shown by graphic displays, a voice generator, and a music generator which expresses his emotional states. AESOPWORLD has an agent-based distributed architecture and also uses frames as semantic representations. It has many modules in common with CHAMELEON although again there is no vision input to AESOPWORLD which uses computer graphics to depict scenes. AESOPWORLD has an extensive planning module but conducts more traditional planning than CHAMELEON's Topsy.

The INTERACT project (Waibel et al. 1996) involves developing MultiModal Human Computer Interfaces including the modalities of speech, gesture and pointing, eye-gaze, lip motion and facial expression, handwriting, face recognition and tracking, and sound localisation. The main concern is with improving recognition accuracies of modality specific component processors as well as developing optimal combinations of multiple input signals to deduce user intent more reliably in cross-modal speech-acts. Wherever possible learning strategies which are mostly connectionist and statistical are developed to ensure scalability and portability to other application domains. INTERACT also uses a frame representation for integrated semantics from gesture and speech and partial hypotheses are developed in terms of partially filled frames. The output of the interpreter is obtained by unifying the information contained in the partial frames. Although Waibel et al. present good work on multimodal interfaces it is not clear that they have produced an integrated platform which can be used for developing multimodal applications.

In summary, there are a number of platforms under development and most of them include two aspects which are characteristic of CHAMELEON: (1) distributed architecture (with blackboard) and (2) semantic representations (as frames). Many include equivalent modules to those in CHAMELEON, some with additional modules (e.g. emotional music) and some lacking modules (e.g. vision, learning).

9.2 Future work

There are a number of avenues for future work with CHAMELEON. We have provided a sample dialogue to be processed by the next prototype that includes examples of (1) spatial relations and (2) anaphoric reference. Another avenue is to improve the dialogue management and modelling capabilities of CHAMELEON so that it can model the user's

beliefs and intentions during the course of a dialogue interaction. The blackboard frame semantics will prove very useful here although, at present, the blackboard is simply a file store of frames and ideas for a more complex design presented in Chapter 3 have yet to be implemented. It is hoped that more complex decision taking can be introduced to operate over semantic representations in the dialogue manager or blackboard using, for example, the HUGIN software tool (Jensen (F.) 1996) based on Bayesian Networks (Jensen (F.V.) 1996). The gesture module will be augmented so that it can handle gestures other than pointing. Topsy will be asked to do more complex learning and processing of input/output from frames. The microphone array has to be integrated into CHAMELEON and set to work. It will be used in applications where there are moving speakers and sound source localisation becomes important but also where there are multiple speakers. All of these ideas will involve augmenting the speech, NLP and gesture processing modules. In general, we want to also focus on the scientific goal of testing varying means of module synchronisation and varying forms of representing semantic and pragmatic information so that they can be optimised. For example, the *blackboard in theory* given in Appendix B treats modules as if they behave in a completely distributed manner with no single coordinating module whilst the present implementation of CHAMELEON, as depicted by the *blackboard in practice* in Appendix C, has a dialogue manager acting as a central coordinator. It would be of interest to implement different behaviours such as these to test the merits of each.

With respect to the IntelliMedia WorkBench application, we can move up to looking at 3D models rather than 2D plans, which will create more work for vision processing. A computer monitor could also display data relevant to the domain such as internet/WWW pages for people or locations referred to. Also, a simulation of the WorkBench could be presented on the screen, or even projected on a big screen so that more people could see what is happening during interaction.

With respect to new applications, we can tackle a whole host (c.f. Chapter 1) such as sign language interpretation, remote presence, MultiMedia CAD, and MultiMedia virtual reality. Application areas of particular interest are *Mobile computing*, *Intelligent Video-Conferencing* and *IntelliMedia information retrieval*.

9.2.1 Mobile computing

The mobile computing aspects of IntelliMedia are particularly relevant. This will enable users to interact with perceptual speech and image data at remote sites and where that data can be integrated and processed at some central source with the possibility of results being relayed back to the user. The increase in bandwidth for wired and wireless networks and the proliferation of hand-held devices (e.g. NOKIA 9000 communicator[1]) and computers (Bruegge and Bennington 1996, Rudnicky et al. 1996, Smailagic and Siewiorek 1996) brings this possibility even closer even with today's separation of information interchange into different data and voice channels. Future introduction of information exchange over a common data/voice channel is expected to further extend mobile computing, and we will

[1]NOKIA 9000 communicator is a trademark of NOKIA.

undoubtedly see the emergence of totally new applications with impressive performance.

Applications of mobile IntelliMedia are numerous including data fusion during emergencies, remote maintenance, remote medical assistance, distance teaching and internet web browsing. One can imagine mobile offices where one can transfer money from/to your bank account, order goods and tickets even while car cruising. The possibility of controlling robots through mobile communications is gaining momentum (Uhlin and Johansson 1996) and will continue to flourish. There are also applications within virtual reality and a feel for these is given in IEEE Spectrum (1997).

The large recent proliferation of conferences on wireless technologies, mobile MultiMedia, computing and networking indicates the surge of interest in this area. There has been rapid convergence of computing and telecommunications technologies in the past few years (IEEE Spectrum 1996). Although there has been great interest in the communication of traditional MultiMedia over wired and wireless networks there has been little development in the area of Intelligent MultiMedia which requires greater bandwidth. For IntelliMedia we need much larger bandwidths, like several 100's of Mb/s, and it is necessary to go to higher carrier frequencies in the millimeter range, where propagation coverage is more limited, and where there are some economic constraints for the moment. There is no doubt, however, that in the beginning of the next century we shall see a major bandwidth expansion for mobility, in the beginning in limited areas. The bandwidths mentioned above should be sufficient for the first generation systems (Roehle 1997).

Wireless mobile computing aspects of our Campus Information System become evident if we consider the user moving away from the WorkBench and walking in the building represented by the 2D plans with a wearable computer (Bruegge and Bennington 1996, Rudnicky et al. 1996, Smailagic and Siewiorek 1996), head-mounted display and even differential global positioning system (D-GPS). Here the system will give directions when the user is moving in the building complex. The wearable computer is online and communicates with the host IntelliMedia Workbench.

9.2.2 IntelliMedia VideoConferencing

Another area where CHAMELEON could be applied is Intelligent VideoConferencing where multiple users can direct cameras through spoken dialogue and gesture. As mentioned in Chapter 1 a miniature version of this idea has already been completed as a student project (Bakman et al. 1997a). Also, such techniques could even be applied in the IntelliMedia WorkBench application where for example multiple speakers (architects) could be planning building and city layout. Other more complex tasks for IntelliMedia include person identification from images and speech.

9.2.3 IntelliMedia information retrieval

SuperinformationhighwayS which have massive stores of information in MultiMedia forms require more intelligent means of information retrieval, where "less" means "more", through

spoken dialogue and other methods. This is and will be a major application area of IntelliMedia (Maybury 1997).

Intelligent MultiMedia will be important in the future of international computing and media development and IntelliMedia 2000+ at Aalborg University, Denmark brings together the necessary ingredients from research, teaching and links to industry to enable its successful implementation. Particularly, we have research groups in spoken dialogue processing and image processing which are the necessary features of this technology. We have a strong commitment to investigate how CHAMELEON can be used for teaching and training. Our CHAMELEON platform and IntelliMedia WorkBench application are ideal for testing integrated signal and symbol processing of language and vision for the future of SuperinformationhighwayS.

Acknowledgements

We take this opportunity to acknowledge funded support from the Faculty of Science and Technology (FaST), Aalborg University, Denmark and from the European Union (EU) under the ESPRIT (OPEN-LTR) Project 24 493. Paul Mc Kevitt would also like to acknowledge the British Engineering and Physical Sciences Research Council (EPSRC) for their generous funded support under grant B/94/AF/1833 for the Integration of Natural Language, Speech and Vision Processing (Advanced Fellow). We would like to acknowledge the Steering Committee for IntelliMedia 2000+: Stig K. Andersen (head, MI), Flemming K. Fink (ESN Masters' coordinator), Flemming B. Frederiksen (head, Institute-8), Erik Granum (head, LIA) and Arne Skou (head, CS), the director of CPK, Jørgen Bach Andersen, Niels Maarbjerg Olesen (deputy Dean) and Finn Kjærsdam (Dean) from the Faculty of Science and Technology, for their invaluable advice and support. Many of them have also read and commented on an earlier draft of this report. We also had stimulating discussions with Anders Bækgaard and Bo Nygaard Bai.

A number of students have done excellent work for CHAMELEON including Lau Bakman and Mads Blidegn (KOM) who set up the laser software drivers, Jørgen B. Nielsen (KOM) who evaluated and downloaded DACS from the University of Bielefeld, and Nicolae Tuns and Thomas Dorf Nielsen (CS) who provided the demo-implementation and initial description of the Topsy module.

Helge Vad (CS) got the Topsy platform running and Bo Nygaard Bai (CPK) helped with specifications for the laser drivers and general systems support. Claus B. Madsen (LIA) participated in discussing the possible demonstrator applications described in Chapter 1 and also read and commented on an earlier draft of this report. Ipke Wachsmuth (University of Bielefeld, Germany) and Kris Thórinsson (LEGO digital, Denmark) are thanked for their useful suggestions on IntelliMedia 2000+ and CHAMELEON while visiting. Thanks to Henrik Benner and Karsten Thygesen for systems and technical support and to Johnna E. Nonboe (graphics, KOM) for help with drawing and tidying some of the pictures herein.

Appendix A

Size of camera lens

The camera is a JAI 2040 colour 1/2" CCD (Charged Coupled Device) camera with auto-gain (see JAI 1996). Here, we show how the size of the camera lens (6mm) is chosen. To understand how the size of the lens can be calculated look at Figure A.1 which shows a pinhole model (Gonzales and Woods 1993) of the camera and the WorkBench table.

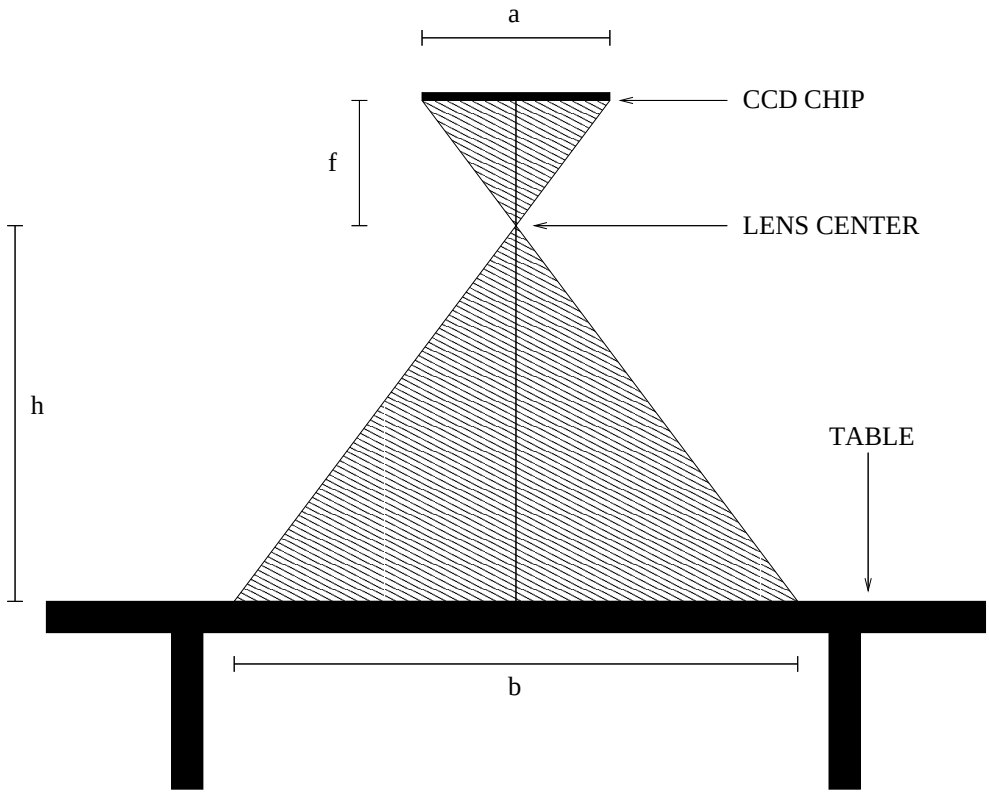

Figure A.1: A pinhole model of camera and WorkBench

The CCD (Charged Coupled Device) chip is shown at the top of the figure and this is where the image (light) is captured. How large an area that can be captured, depends on

the size of the chip a, the focal length, f, and the distance from the camera to the table, h. The focal length is the distance from the CCD chip to the center of the lens and this is what is referred to as the *size* of the lens. The smaller this distance is, the larger an area can be captured. But as f decreases, the resolution also decreases. So f should not be chosen too small.

Due to the relations between the triangles in figure A.1 the following relations can be stated.

$$\frac{a/2}{f} = \frac{b/2}{h} \quad \Leftrightarrow \tag{A.1}$$

$$\frac{a}{f} = \frac{b}{h} \quad \Leftrightarrow \tag{A.2}$$

$$f = \frac{a \cdot h}{b} \quad \Leftrightarrow \tag{A.3}$$

$$b = \frac{a \cdot h}{f} \tag{A.4}$$

The maximum size of the lens can now be calculated. Since neither the CCD chip or the table are squared, two cases must be evaluated; one for the x-direction and one for the y-direction. To do so, look at figure A.2 where the relation between the CCD chip and the table is shown.

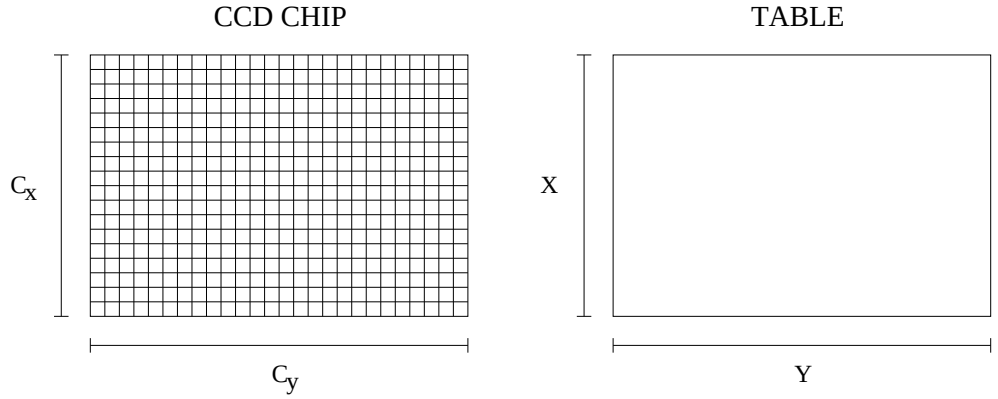

Figure A.2: Parameters defining CCD chip and table

First the calculation is done in the x-direction where the parameter of the CCD chip is $C_x = 0.66$cm and the parameter of the table is $x = 185$cm. Using equation A.3 yields $f_x = 9.3$mm Next the calculation is done in the y-direction where the parameter of the CCD chip is $C_y = 0.44$cm and the parameter of the table is $y = 120$cm. Using equation A.3 yields $f_y = 9.5$mm This means that the lens should be smaller than 9.3mm. A normal lens is very unlinear in the outer regions and therefore it is normally a good idea to chose the lens a bit smaller than required to avoid this problem. Also, there should be room for

movements of the table without forced the camera to be moved. Therefore, a 6mm lens is chosen. When inserting this choice in equation A.4 the new workspace is $286 * 191$cm.

Appendix B

Blackboard in theory

Here we derive all the frames appearing on the blackboard for two examples: (1) "Point to Hanne's office" (instruction), and (2) "Whose office is this?" + [pointing] (exophoric/deictic reference).

There are input, output and integration frames (F-in, F-out, F-int), input and output gestures (G-in, G-out) and input and output utterances (U-in, U-out). Input modules are SPEECH-RECOGNISER (U-in) and GESTURE (G-in). Output modules are LASER (G-out) and SPEECH-SYNTHESIZER (U-out). Most modules give and take frames to/from the blackboard database and process them (F-int).

We choose to have modules interacting in a completely distributed manner with no single coordinating module. The actual present implementation of CHAMELEON has a dialogue manager which acts as a central coordinator. Although we show the various modules acting in a given sequence here, module processing and frames may not necessarily run in this order. The frames given are placed on the blackboard as they are produced and processed.

Example 1: giving an instruction

USER(U-in): Point to Hanne's office

PROCESSING(1):
SPEECH-RECOGNISER:
(1) wakes up when it detects registering of U-in
(2) maps U-in into F-in
(3) places and registers F-in on blackboard:

FRAME(F-in)(1):
[SPEECH-RECOGNISER
UTTERANCE: (Point to Hanne's office)
INTENTION: instruction!

TIME: timestamp]

PROCESSING(2):
NLP:
(1) wakes up when it detects registering of F-in
(2) maps F-in into F-int
(3) places and registers F-int on blackboard:

FRAME(F-int)(1):
[NLP
INTENTION: instruction! (pointing)
LOCATION: office (tenant Hanne) (coordinates (X, Y))
TIME: timestamp]

PROCESSING(3):
DOMAIN-MODEL:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees its from NLP
(3) produces updated F-int (coordinates)
(4) places and registers updated F-int on blackboard:

FRAME(F-int)(2):
[DOMAIN-MODEL
INTENTION: instruction! (pointing)
LOCATION: office (owner Hanne) (coordinates (5, 2))
TIME: timestamp]

PROCESSING(4):
NLP:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees it's from DOMAIN-MODEL
(3) produces updated F-int (intention + utterance)
(4) places and registers updated F-int on blackboard:

FRAME(F-int)(3):
[NLP
INTENTION: description (pointing)
LOCATION: office (owner Hanne) (coordinates (5, 2))
UTTERANCE: (This is Hanne's office)
TIME: timestamp]

PROCESSING(5):
LASER:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees it's from DOMAIN-MODEL

(3) produces F-out (pruning + registering)
(4) places and registers F-out on blackboard:

FRAME(F-out)(1):
[LASER
INTENTION: description (pointing)
LOCATION: coordinates (5, 2)
TIME: timestamp]

PROCESSING(6):
SPEECH-SYNTHESIZER:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees it's from NLP
(3) produces F-out (pruning + registering)
places and registers F-out on blackboard:

FRAME(F-out)(2):
[SPEECH-SYNTHESIZER
INTENTION: description
UTTERANCE: (This is Hanne's office)
TIME: timestamp]

PROCESSING(7):
DIALOGUE MANAGER:
(1) wakes up when it detects registering of F-out and F-out
(2) reads F-out and F-out and sees they are from
LASER and SPEECH-SYNTHESIZER
(3) dials and fires LASER and SPEECH-SYNTHESIZER
in a rhythmic way (synchronized)
(1) LASER reads its own F-out and fires G-out
(2) SPEECH-SYNTHESIZER reads its own F-out and fires U-out

CHAMELEON(G-out): [points]
CHAMELEON(U-out): This is Hanne's office.

Example 2: deictic reference

USER(G-in,U-in): [points]
Whose office is this?

PROCESSING(1):
SPEECH-RECOGNISER:
(1) wakes up when it detects registering of U-in
(2) maps U-in into F-in

(3) places and registers F-in on blackboard

FRAME(F-in)(1):
[SPEECH-RECOGNISER
UTTERANCE: (Whose office is this ?)
INTENTION: query?
TIME: timestamp]

PROCESSING(2):
NLP:
(1) wakes up when it detects registering of F-in
(2) maps F-in into F-int
(3) places and registers F-int on blackboard:

FRAME(F-int)(1):
[NLP
INTENTION: query? (who)
LOCATION: office (tenant person) (coordinates (X, Y))
REFERENT: this
TIME: timestamp]

PROCESSING(3):
DOMAIN-MODEL:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees its from NLP
(3) cannot update F-int as doesn't have a name or coordinates
(4) goes back to sleep

PROCESSING(4):
GESTURE:
(1) wakes up when it detects registering of G-in
(2) maps G-in into F-in
(3) places and registers F-in on blackboard

FRAME(F-in)(2):
[GESTURE
GESTURE: coordinates (3, 2)
INTENTION: pointing
TIME: timestamp]

PROCESSING(5):
DIALOGUE MANAGER:
(1) wakes up when it detects registering of F-in(1) and F-in(2)
(2) reads F-in(1) and F-in(2) and
sees they are from SPEECH-RECOGNISER and GESTURE

that they have same/close timestamp,
there is a query? (with referent) + pointing,
in a rhythmic way (synchronized)
(3) dials and fires NLP to read GESTURE

PROCESSING(6):
NLP:
(1) woken up by DIALOGUE-MANAGER and reads F-in(2)
(2) sees F-in(2) is from GESTURE
(3) determines referent of "this" to be (coordinates)
(4) produces updated F-int (coordinates)
(5) places and registers updated F-int on blackboard:

FRAME(F-int)(2):
[NLP
INTENTION: query? (who)
LOCATION: office (tenant person) (coordinates (3, 2))
REFERENT: this
TIME: timestamp]

PROCESSING(7):
DOMAIN-MODEL:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees its from NLP
(3) produces updated F-int (tenant)
(4) places and registers updated F-int on blackboard:

FRAME(F-int)(3):
[NLP
INTENTION: query? (who)
LOCATION: office (tenant Ipke) (coordinates (3, 2))
REFERENT: this
TIME: timestamp]

PROCESSING(8):
NLP:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees it's from DOMAIN-MODEL
(3) produces updated F-int (intention + utterance)
(4) places and registers updated F-int on blackboard:

FRAME(F-int)(4):
[NLP
INTENTION: declarative (who)
LOCATION: office (tenant Ipke) (coordinates (3, 2))

REFERENT: this
UTTERANCE: (This is Ipke's office)
TIME: timestamp]

PROCESSING(9):
LASER:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees it's from DOMAIN-MODEL
(3) produces F-out (pruning + registering)
(4) places and registers F-out on blackboard:

FRAME(F-out)(1):
[LASER
INTENTION: description (pointing)
LOCATION: coordinates (3, 2)
TIME: timestamp]

PROCESSING(10):
SPEECH-SYNTHESIZER:
(1) wakes up when it detects registering of F-int
(2) reads F-int and sees it's from NLP
(3) produces F-out (pruning + registering)
places and registers F-out on blackboard:

FRAME(F-out)(2):
[SPEECH-SYNTHESIZER
INTENTION: description
UTTERANCE: (This is Ipke's office)
TIME: timestamp]

PROCESSING(11):
DIALOGUE-MANAGER:
(1) wakes up when it detects registering of F-out(1) and F-out(2)
(2) reads F-out(1) and F-out(2) and sees they are from
LASER and SPEECH-SYNTHESIZER
(3) dials and fires LASER and SPEECH-SYNTHESIZER
in a rhythmic way (synchronized)
(1) LASER reads G-out and fires G-out
(2) SPEECH-SYNTHESIZER reads U-out and fires U-out

CHAMELEON(G-out): [points]
CHAMELEON(U-out): This is Ipke's office.

Appendix C

Blackboard in practice

Here we show the complete blackboard (with all frames) as produced exactly by CHAMELEON for the example dialogue given in Chapter 2, Section 2.2. Note that unlike the blackboard in theory above (Appendix B) the present implementation of CHAMELEON has a dialogue manager which acts as a central coordinator.

```
Received: nlp(intention(instruction(pointing)),location(person(tb),type(office)),
time(889524794))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(point(coordinates(249,623))),
speech_synthesizer(utterance("This is Toms office"))))
Calling laser: laser(point(coordinates(249,623)))
Calling speech_synthesizer: speech_synthesizer(utterance("This is Toms office"))
```

```
Received: nlp(intention(instruction(pointing)),location(person(tbm),type(office)),
time(889524818))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(point(coordinates(278,623))),
speech_synthesizer(utterance("This is Thomass office"))))
Calling laser: laser(point(coordinates(278,623)))
Calling speech_synthesizer: speech_synthesizer(utterance("This is Thomass office"))
```

```
Received: nlp(intention(query(where)),location(place(a2_221)),
time(889524831))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(point(coordinates(132,500))),
speech_synthesizer(utterance("computer room is here"))))
Calling laser: laser(point(coordinates(132,500)))
Calling speech_synthesizer: speech_synthesizer(utterance("computer room is here"))
```

```
Received: nlp(intention(query(who)),location(this($Deixis),type(office)),
time(889524864))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(point(coordinates(658,546))),
speech_synthesizer(utterance("This is not an office, this is instrument repair"))))
Calling laser: laser(point(coordinates(658,546)))
Calling speech_synthesizer:
speech_synthesizer(utterance("This is not an office, this is instrument repair"))


Received: nlp(intention(query(who)),location(this($Deixis),type(office)),
time(889524885))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(point(coordinates(223,568))),
speech_synthesizer(utterance("This is Pauls office"))))
Calling laser: laser(point(coordinates(223,568)))
Calling speech_synthesizer: speech_synthesizer(utterance("This is Pauls office"))


Received: nlp(intention(instruction(show_route)),source(location(person(lbl),
type(office))),
destination(location(person(hg),type(office))),time(889524919))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(route(coordinates(278,585,278,603,249,
603,220,603,197,603,197,623))),
speech_synthesizer(
utterance("This is the route from Lars Bos office to Hannes office"))))
Calling laser:
laser(route(coordinates(278,585,278,603,249,603,220,603,197,603,197,623)))
Calling speech_synthesizer:
speech_synthesizer(
utterance("This is the route from Lars Bos office to Hannes office"))


Received: nlp(intention(instruction(show_route)),source(location(person(pmck),
type(office))),destination(location(place(a2_105))),time(889524942))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(route(coordinates(174,453,153,453,153,
481,153,500,153,510,153,540,153,569,153,599,153,603,184,603,197,603,220,603,
249,603,278,603,307,603,330,603,330,655,354,655,911,655,884,655,884,603,810,
603,759,603,717,603,717,570,696,570))),
speech_synthesizer(
utterance("This is the route from Pauls office to instrument repair"))))
```

```
Calling laser: laser(route(coordinates(174,453,153,453,153,481,153,500,153,
510,153,540,153,569,153,599,153,603,184,603,197,603,220,603,249,603,278,603,
307,603,330,603,330,655,354,655,911,655,884,655,884,603,810,603,759,603,717,
603,717,570,696,570)))
Calling speech_synthesizer:
speech_synthesizer(
utterance("This is the route from Pauls office to instrument repair"))


Received: nlp(intention(instruction(pointing)),location(person(pd),type(office)),
time(889524958))
which is passed on to dialog_manager
Received: dialog_manager(output(laser(point(coordinates(220,585))),
speech_synthesizer(utterance("This is Pauls office"))))
```

Appendix D

Syntax of frames

The following BNF grammar defines a predicate-argument syntax for the form of messages (frames) appearing on CHAMELEON's implemented blackboard.

```
FRAME           ::= PREDICATE

PREDICATE       ::= identifier(ARGUMENTS)

ARGUMENTS       ::= ARGUMENT
                  | ARGUMENTS, ARGUMENT

ARGUMENT        ::= CONSTANT
                  | VARIABLE
                  | PREDICATE

CONSTANT        ::= identifier
                  | integer
                  | string

VARIABLE        ::= $identifier
```

FRAME acts as start symbol, CAPITAL symbols are non-terminals, and terminals are lower-case or one of the four symbols `( ) ,` and `$`. An *identifier* starts with a letter that can be followed by any number of letters, digits or `_`, an *integer* consists of a sequence of digits and a *string* is anything delimited by two `"`'s. Thus the *alphabet* consists of the letters, the digits and the symbols `( ) , _` and `$`. A parser has been written in C which can parse the frames using this BNF definition.

Appendix E

Camera calibration

Here, a detailed description of camera calibration for CHAMELEON is given.

E.1 A homogeneous representation

Intuitively, the calibration can be explained as finding the translation and rotation differences between the 2D plan coordinate system and the camera coordinate system. When dealing with relations between an image and a world coordinate system, a transformation equation between the two can be defines as

$$\mathbf{c_h} = \mathbf{A} \cdot \mathbf{w_h} \tag{E.1}$$

where $\mathbf{c_h}$ holds information about the image point (x_i, y_i) and $\mathbf{w_h}$ holds information about the world point (X_w, Y_w, Z_w). $\mathbf{A}$ is the transformation matrix which maps from one coordinate system to the other. It holds information about the rotation, translation and perspective transformation through the camera lens (Gonzalez and Woods 1993).

Homogeneous coordinates are used to represent both image points and world points. This is done in order to make $\mathbf{A}$ a square matrix facilitating calculations. However, this means that the transformation is not done directly between the image point and the world point, but instead between the homogeneous world point and the homogeneous and perspective transformed image point (Gonzalez and Woods 1993) as shown here,

$$\begin{bmatrix} c_{h1} \\ c_{h2} \\ c_{h3} \\ c_{h4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} kX_w \\ kY_w \\ kZ_w \\ k \end{bmatrix}$$

where,

$$\frac{c_{h1}}{c_{h4}} = x_i \quad \text{(E.2)}$$

$$\frac{c_{h2}}{c_{h4}} = y_i \quad \text{(E.3)}$$

Substituting E.2 and E.3 into the homogeneous equation system and expanding the matrix product for the first, second and third row with $k = 1$ yields,

$$x_i c_{h4} = a_{11}X_w + a_{12}Y_w + a_{13}Z_w + a_{14} \quad \text{(E.4)}$$

$$y_i c_{h4} = a_{21}X_w + a_{22}Y_w + a_{23}Z_w + a_{24} \quad \text{(E.5)}$$

$$c_{h4} = a_{41}X_w + a_{42}Y_w + a_{43}Z_w + a_{44} \quad \text{(E.6)}$$

The world coordinate system is defined to have its X-Y-plane parallel to the physical 2D plan and $Z_w = 0$ for all world points located on the 2D plan. Using this decision and substitute E.6 into E.4 and E.5 respectively yields,

$$a_{11}X_w + a_{12}Y_w + a_{14} - x_i a_{41}X_w - x_i a_{42}Y_w - x_i a_{44} = 0 \quad \text{(E.7)}$$

$$a_{21}X_w + a_{22}Y_w + a_{24} - y_i a_{41}X_w - y_i a_{42}Y_w - y_i a_{44} = 0 \quad \text{(E.8)}$$

E.2 Calculating coefficients

The coefficients in **A** (homogeneous transformation) need to be calculated before the actual mapping function between the two coordinate systems can be derived. The problem is now to solve the two equations; E.7 and E.8 with respect to the coefficients, a_{xy}. A world point and the image point this is mapped into can easily be obtained, but it is still not possible to solve the equations directly since the problem is underdetermined, two equations and nine unknowns. The problem is solved by adding more information without increasing the number of unknowns. To avoid the zero-solution it is chosen to fix one of the coefficients, $a_{24} = -1$. Now three other pairs of equations like E.7 and E.8 are set up with different values of (x_i, y_i) and (X_w, Y_w)

$$
\begin{aligned}
a_{11}X_{w1} + a_{12}Y_{w1} + a_{14} - x_{i1}a_{41}X_{w1} - x_{i1}a_{42}Y_{w1} - x_{i1}a_{44} &= 0 \\
a_{21}X_{w1} + a_{22}Y_{w1} + a_{24} - y_{i1}a_{41}X_{w1} - y_{i1}a_{42}Y_{w1} - y_{i1}a_{44} &= 0 \\
\\
a_{11}X_{w2} + a_{12}Y_{w2} + a_{14} - x_{i2}a_{41}X_{w2} - x_{i2}a_{42}Y_{w2} - x_{i2}a_{44} &= 0 \\
a_{21}X_{w2} + a_{22}Y_{w2} + a_{24} - y_{i2}a_{41}X_{w2} - y_{i2}a_{42}Y_{w2} - y_{i2}a_{44} &= 0 \\
\\
a_{11}X_{w3} + a_{12}Y_{w3} + a_{14} - x_{i3}a_{41}X_{w3} - x_{i3}a_{42}Y_{w3} - x_{i3}a_{44} &= 0 \\
a_{21}X_{w3} + a_{22}Y_{w3} + a_{24} - y_{i3}a_{41}X_{w3} - y_{i3}a_{42}Y_{w3} - y_{i3}a_{44} &= 0 \\
\\
a_{11}X_{w4} + a_{12}Y_{w4} + a_{14} - x_{i4}a_{41}X_{w4} - x_{i4}a_{42}Y_{w4} - x_{i4}a_{44} &= 0 \\
a_{21}X_{w4} + a_{22}Y_{w4} + a_{24} - y_{i4}a_{41}X_{w4} - y_{i4}a_{42}Y_{w4} - y_{i4}a_{44} &= 0
\end{aligned}
$$

The new indexes indicate coherency, meaning that (X_{w1}, Y_{w1}) is mapped into the image coordinate system as (x_{i1}, y_{i1}). These four coherent pairs can be measured and therefore the problem is now solvable since it consists of eight equations with eight unknowns. The equations are rearranged into a linear system $\mathbf{u} = \mathbf{H} \cdot \mathbf{b}$

$$
\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} =
\begin{bmatrix}
X_{w1} & Y_{w1} & 0 & 0 & -X_{w1}x_{i1} & -Y_{w1}x_{i1} & -x_{i1} & 1 \\
0 & 0 & X_{w1} & Y_{w1} & -X_{w1}y_{i1} & -Y_{w1}y_{i1} & -y_{i1} & 0 \\
X_{w2} & Y_{w2} & 0 & 0 & -X_{w2}x_{i2} & -Y_{w2}x_{i2} & -x_{i2} & 1 \\
0 & 0 & X_{w2} & Y_{w2} & -X_{w2}y_{i2} & -Y_{w2}y_{i2} & -y_{i2} & 0 \\
X_{w3} & Y_{w3} & 0 & 0 & -X_{w3}x_{i3} & -Y_{w3}x_{i3} & -x_{i3} & 1 \\
0 & 0 & X_{w3} & Y_{w3} & -X_{w3}y_{i3} & -Y_{w3}y_{i3} & -y_{i3} & 0 \\
X_{w4} & Y_{w4} & 0 & 0 & -X_{w4}x_{i4} & -Y_{w4}x_{i4} & -x_{i4} & 1 \\
0 & 0 & X_{w4} & Y_{w4} & -X_{w4}y_{i4} & -Y_{w4}y_{i4} & -y_{i4} & 0
\end{bmatrix} \cdot
\begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ a_{41} \\ a_{42} \\ a_{44} \\ a_{14} \end{bmatrix}
$$

The coefficients of $\mathbf{A}$ can now be found as

$$\mathbf{b} = \mathbf{H}^{-1} \cdot \mathbf{u} \tag{E.9}$$

Many numerical techniques exist for finding an optimal solution for $\mathbf{b}$. It is chosen to use the Singular Value Decomposition method since this is a stable method which is easy to implement (Press et al. 1990).

E.3 Finding the corresponding pairs

Before equation E.9 can be solved the four pairs of corresponding image points and world points need to be found. As already mentioned the world coordinate system is chosen to

be located on top of the 2D plan. The origin of the world coordinate system is placed in the upper left corner of the 2D plan having the X and Y-axis following the edges of the 2D plan and the Z-axis equal to zero in the plane spanned by the 2D plan. In this way the world coordinate system is defined by the location and orientation of the 2D plan.

The four image points should be easily located landmarks in order to make the calibration robust. It is chosen to use the corners of the plan since they are present anyway. This also means that no artificial markers need be introduced. The position of the four corners is measured (using a ruler) in the world coordinate system. Since the 2D plan defines the world coordinate system and since the plan is a rigid object, these four measurements only need to be done once and then the results can be reused every time the camera is calibrated.

E.3.1 Finding the image points

The camera will not be moved very often but the 2D plan might. Therefore the calibration should be done automatically. The algorithm for finding the four corner points is designed to be independent of how the 2D plan is oriented on the table, as long as it is not entering area 2. Two versions of this algorithm are implemented, each having a different assumption of how the 2D plan is oriented. The algorithms are processed in parallel and the one which comes up with the best result will determine the location of the four corner points. The two orientation assumptions for the two versions are shown in Figure E.1.

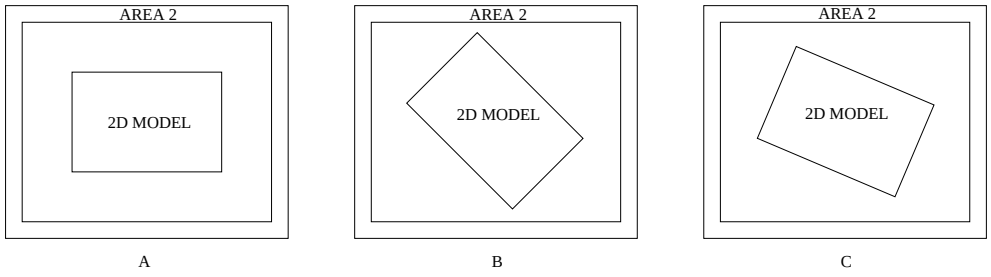

Figure E.1: A) first assumption; B) second assumption; C) in-between situation

The first version of the algorithm assumes that the 2D plan is located as shown in Figure E.1.a, while the second version assumes Figure E.1.b. The closer the 2D plan is to one of the figures the more unambiguous the result will be. When the 2D plan is oriented in between the two assumptions, as shown in Figure E.1.c, it is random as to which version delivers the best result. However, the results should still be correct, but the uncertainty grows the further the orientation of the 2D plan gets from the situations shown in Figure E.1.a and E.1.b.

Independent of which version of the algorithm is used, the method is the same. Each corner is found separately but using the same technique which will be explained for one corner of assumption a. First a search is carried out towards the corner as shown in Figure E.2.a.

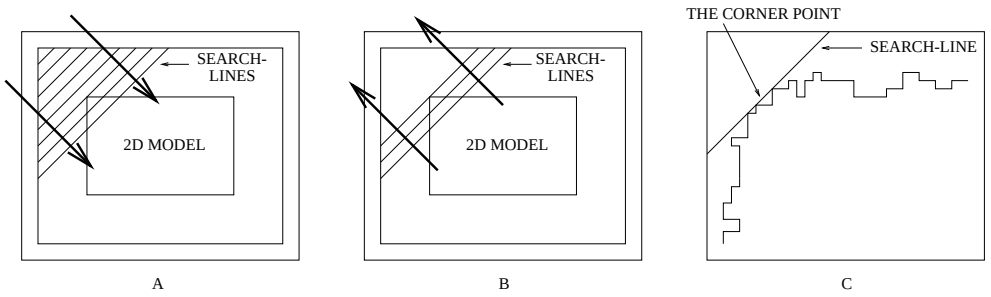

Figure E.2: A) first search for one corner; B) second search for one corner; C) close up of corner and found corner point

The search is done in lines perpendicular to the assumed orientation of the 2D plan and stops when a predefined number of "white" pixels (compared to the "dark" background) are found in one line. The idea is to make this predefined number so high that the influence of noise is removed, meaning that this search does not stop until it is absolutely sure that it has found the 2D plan and not just some random noise. Hereafter, the second search as shown in Figure E.2.b, is carried out. It also searches in lines but in the opposite direction. It stops when no more "white" pixels can be found in a line and concludes that the corner must be in the previous search line. The actual corner point is found as the average position of all "white" pixels in this line.

It might come as a surprise to the reader that a corner has to be found in this way. The reason for this can be seen in Figure E.2.c where a close up of a corner is shown. The problem is that a corner is far from well defined when you look at it on pixel level and hence some kind of averaging must be done.

E.3.2 Evaluation of corner points

Both versions of the algorithm come up with four image points which are believed to be the correct corner points. But how do we figure out which set of points is the correct one? For each version of the algorithm the following is done. The distances between every point is calculated yielding six numbers which are sorted according to their values

$$\mathbf{D} = [d_1 \; d_2 \; d_3 \; d_4 \; d_5 \; d_6]^T \tag{E.10}$$

where d_6 is the highest value.

If the 2D plan was a perfect rectangle and the camera/framegrabber was ideal then $d_1 = d_2$, $d_3 = d_4$ and $d_5 = d_6$. Even though nothing is ideal the following equation, ψ, will still give a rather good indication of how good the data matches a rectangle

$$\psi = |d_1 - d_2| + |d_3 - d_4| + |d_5 - d_6| \tag{E.11}$$

After implementing this it was noticed that ψ can be rather small without representing the 2D model (or a rectangle) if the corner candidates are found in certain unfavourable positions, for example, in the same position. This problem is solved by using knowledge about the shape of the 2D plan. Since it is a rectangle $d_1 < d_6$. Equation E.11 is now expanded into

$$\psi = \begin{cases} |d_1 - d_2| + |d_3 - d_4| + |d_5 - d_6| & \text{if } d_1 \cdot t < d_6 \\ -42 & \text{otherwise} \end{cases} \tag{E.12}$$

where t is found imperial to 1.3. For each version of the algorithm ψ is calculated and the one with the lowest positive value is the version which has found the correct corner points.

E.4 From image points to world points

In this section the two equations which are used to map an image point (x_i, y_i) into a world point (X_w, Y_w) are derived. To make the calculations easier to follow, equation E.7 and E.8 are rewritten

$$\begin{aligned} X_w(a_{11} - x_i a_{41}) + Y_w(a_{12} - x_i a_{42}) + (a_{14} - x_i a_{44}) &= 0 \Rightarrow & \text{(E.13)} \\ X_w \alpha_x + Y_w \beta_x + \gamma_x &= 0 & \text{(E.14)} \end{aligned}$$

$$\begin{aligned} X_w(a_{21} - y_i a_{41}) + Y_w(a_{22} - y_i a_{42}) + (a_{24} - y_i a_{44}) &= 0 \Rightarrow & \text{(E.15)} \\ X_w \alpha_y + Y_w \beta_y + \gamma_y &= 0 & \text{(E.16)} \end{aligned}$$

Isolating X_w in E.14 yields

$$X_w = \frac{-Y_w \beta_x - \gamma_x}{\alpha_x} \tag{E.17}$$

Inserting X_w in E.16 yields

$$\begin{aligned} 0 &= (\frac{-Y_w \beta_x - \gamma_x}{\alpha_x}) \cdot \alpha_y + Y_w \beta_y + \gamma_y \Rightarrow & \text{(E.18)} \\ 0 &= Y_w \cdot (\beta_y - \frac{\beta_x \alpha_y}{\alpha_x}) + \gamma_y - \frac{\gamma_x \alpha_y}{\alpha_x} \Rightarrow & \text{(E.19)} \\ Y_w &= \frac{\frac{\gamma_x \alpha_y}{\alpha_x} - \gamma_y}{\beta_y - \frac{\beta_x \alpha_y}{\alpha_x}} \Rightarrow & \text{(E.20)} \\ Y_w &= \frac{\gamma_x \alpha_y - \gamma_y \alpha_x}{\beta_y \alpha_x - \beta_x \alpha_y} & \text{(E.21)} \end{aligned}$$

X_w can now be found as

$$\begin{aligned}
X_w &= \frac{-Y_w\beta_x - \gamma_x}{\alpha_x} \Rightarrow \\
X_w &= \frac{-(\frac{\gamma_x\alpha_y - \gamma_y\alpha_x}{\beta_y\alpha_x - \beta_x\alpha_y}) \cdot \beta_x - \gamma_x}{\alpha_x} \Rightarrow \\
X_w &= \frac{-\beta_x \cdot (\gamma_x\alpha_y - \gamma_y\alpha_x) - \gamma_x \cdot (\beta_y\alpha_x - \beta_x\alpha_y)}{\alpha_x \cdot (\beta_y\alpha_x - \beta_x\alpha_y)} \Rightarrow \\
X_w &= \frac{-\beta_x\gamma_x\alpha_y + \beta_x\gamma_y\alpha_x - \gamma_x\beta_y\alpha_x + \gamma_x\beta_x\alpha_y}{\alpha_x\beta_y\alpha_x - \alpha_x\beta_x\alpha_y}
\end{aligned}$$

E.5 From world points to image points

In this section the two equations which are used to map a world point (X_w, Y_w) into an image point (x_i, y_i) are derived. The image point is simply found by isolating x_i in equation E.7 and y_i in equation E.8

$$\begin{aligned}
0 &= -a_{11}X_w - a_{12}Y_w - a_{14} + x_i a_{41}X_w + x_i a_{42}Y_w + x_i a_{44} \Rightarrow \\
0 &= x_i(a_{41}X_w + a_{42}Y_w + a_{44}) - (a_{11}X_w + a_{12}Y_w + a_{14}) \Rightarrow \\
x_i &= \frac{a_{11}X_w + a_{12}Y_w + a_{14}}{a_{41}X_w + a_{42}Y_w + a_{44}}
\end{aligned}$$

$$\begin{aligned}
0 &= -a_{21}X_w - a_{22}Y_w - a_{24} + y_i a_{41}X_w + y_i a_{42}Y_w + y_i a_{44} \Rightarrow \\
0 &= y_i(a_{41}X_w + a_{42}Y_w + a_{44}) - (a_{21}X_w + a_{22}Y_w + a_{24}) \Rightarrow \\
y_i &= \frac{a_{21}X_w + a_{22}Y_w + a_{24}}{a_{41}X_w + a_{42}Y_w + a_{44}}
\end{aligned}$$

E.6 Summary

In this appendix camera calibration for CHAMELEON and the IntelliMedia WorkBench have been described. The purpose of the calibration is to find a transformation between a world point and an image point. A homogeneous transformation system is set up where four world points and the corresponding image points are needed. The world points are defined to be the corners of the plan which can be measured directly on the 2D plan. The corresponding image points are found automatically using a three level search method. When the homogeneous transformation matrix is calculated, the actual transformation between a world point and an image point (and visa versa) are derived.

The system is now self-calibrating with respect to the camera and the 2D plan. Since the 2D plan (or the table it is located on) is often moved, the calibration is done every

time the tracking module is started up and last for about 5 seconds. Of course nothing can interact with the plan in this period.

Appendix F

NLP grammar

Here, we give the axioms, lexical, syntactic and semantic rules for the NLP grammar.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                         MMUI.APS                                  %%
%%            Multimedia Multimodal User Interface                   %%
%%              augmented phrase structure grammar                   %%
%%                                                                   %%
%%      Version: 1.3                                                 %%
%%      Date   : Jan 98                                              %%
%%      Version History:                                             %%
%%            v. 0.0 based on pure syntactic parsing and percolation %%
%%                   of semantic values. Output a simple feature set %%
%%                   equivalent to the format planned for REWARD.    %%
%%                   Designers: bakman & blidegn.                    %%
%%            v. 1.0 based on syntactic parsing and semantic mapping,%%
%%                   Output equivalent to the "semantic represen-    %%
%%                   tations" defined in the AB-platform und used in %%
%%                   the P1 flight ticket reservation system.        %%
%%                   Designer: tb                                    %%
%%            v. 1.1 based on syntactic parsing and mapping into struc- %%
%%                   equivalent to the predicate-argument BNF defined %%
%%                   for the MMUI. APS. The method utilizes new fasci- %%
%%                   lities built into the NL parser (dynamic genera- %%
%%                   tion of semantic trees as opposed to the static %%
%%                   structures of version 1.0).                     %%
%%            v. 1.2 added persons and places                        %%
%%            v. 1.3 The syntax of semantic mapping rules has been   %%
%%                   slightly modified (glue nodes)                  %%
%%                                                                   %%
%%      Tom Broendsted                                               %%
%%                                                                   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                  AXIOMS                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%## English
axiom1 = [{cat=s}].


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               LEXICAL RULES                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Noun: Persons
%%macros
#firstname={cat=proper,case=no,semtype=person,surname=no}.
#surname={cat=proper,case=no,semtype=person,surname=yes}.
#firstname_gen={cat=proper,case=gen,semtype=person,surname=no}.
#surname_gen={cat=proper,case=gen,semtype=person,surname=yes}.

{lex=ove,phon="ow v ax sp",id=oa,#firstname}.
{lex=andersen,phon="ae n d er s ax n sp",id=oa,#surname}.
{lex=oves,phon="ow v ax z sp",id=oa,#firstname_gen}.
{lex=andersens,phon="b aa k ae n d er s ax n z sp",id=oa,#surname_gen}.

{lex=jorgen,phon="y er ng sp",id=jba,#firstname}.
{lex=bach_andersen,phon="b aa k ae n d er s ax n sp",id=jba,#surname}.
{lex=jorgens,phon="y er ng z sp",id=jba,#firstname_gen}.
{lex=bach_andersens,phon="b aa k ae n d er s ax n z sp",id=jba,#surname_gen}.

{lex=bo,phon="b ow sp",id=bai,#firstname}.
{lex=bai,phon="b ay sp",id=bai,#surname}.
{lex=bos,phon="b ow z sp",id=bai,#firstname_gen}.
{lex=bais,phon="b ay z sp",id=bai,#surname_gen}.

{lex=tom,phon="t oh m sp",id=tb,#firstname}.
{lex=broendsted,phon="b r ao n s d ax dh sp",id=tb,#surname}.
{lex=toms,phon="t oh m z sp",id=tb,#firstname_gen}.
{lex=broendsteds,phon="b r ao n s d ax dh z sp",id=tb,#surname_gen}.

{lex=paul,phon="p ao l s|p oh l sp",id=pd,#firstname}.
{lex=dalsgaard,phon="d ae l s g ao sp",id=pd,#surname}.
{lex=pauls,phon="p ao l z sp|p oh l z sp",id=pd,#firstname_gen}.
{lex=dalsgaards,phon="d ae l s g ao z sp",id=pd,#surname_gen}.

/*
```

```
{lex=peter,phon="",id=ped,#firstname}.
{lex=dissing,phon="",id=ped,#surname}.
{lex=peters,phon="",id=ped,#firstname_gen}.
{lex=dissing,phon="",id=ped,#surname_gen}.
*/
{lex=hanne,phon="hh ae n ih sp",id=hg,#firstname}.
{lex=gade,phon="g ae dh er sp",id=hg,#surname}.
{lex=hannes,phon="hh ae n ih z sp",id=hg,#firstname_gen}.
{lex=gades,phon="g ae dh er z sp",id=hg,#surname_gen}.
/*
{lex=soren,phon="",id=shj,#firstname}.
{lex=holdt_jensen,phon="",id=shj,#surname}.
{lex=sorens,phon="",id=shj,#firstname_gen}.
{lex=holdt_jensens,phon="",id=shj,#surname_gen}.

{lex=soren_peter,phon="",id=spj,#firstname}.
{lex=jacobsen,phon="",id=spj,#surname}.
{lex=soren_peters,phon="",id=spj,#firstname_gen}.
{lex=jacobsens,phon="",id=spj,#surname_gen}.

{lex=jesper,phon="",id=jje,#firstname}.
{lex=jensen,phon="",id=jje,#surname}.
{lex=jespers,phon="",id=jje,#firstname_gen}.
{lex=jensens,phon="",id=jje,#surname_gen}.
*/
{lex=lars_bo,phon="l aa r z b ow sp",id=lbl,#firstname}.
{lex=larsen,phon="l aa r s ih n sp|l aa s ih n sp",id=lbl,#surname}.
{lex=lars_bos,phon="l aa r z b ow s sp",id=lbl,#firstname_gen}.
{lex=larsens,phon="l aa r s ih n z sp|l aa s ih n z sp",id=lbl,#surname_gen}.

{lex=borge,phon="b ao g ax sp|b ao r g ax sp",id=bli,#firstname}.
{lex=lindberg,phon="l ih n d b er g sp",id=bli,#surname}.
{lex=borges,phon="b ao g ax z sp|b ao r g ax z sp",id=bli,#firstname_gen}.
{lex=lindbergs,phon="l ih n d b er g z sp",id=bli,#surname_gen}.

{lex=paul,phon="p ao l sp|p oh l sp",id=pmck,#firstname}.
{lex=mckevitt,phon="m aa k eh v iy t sp",id=pmck,#surname}.
{lex=pauls,phon="p ao l z sp|p oh l z sp",id=pmck,#firstname_gen}.
{lex=mckevitts,phon="m aa k eh v iy t z sp",id=pmck,#surname_gen}.

{lex=thomas,phon="t oh m ax s sp",id=tbm,#firstname}.
{lex=moeslund,phon="m uh s l ow n d sp",id=tbm,#surname}.
{lex=thomas,phon="t oh m ax s sp",id=tbm,#firstname_gen}.
{lex=moeslunds,phon="m uh s l ow n d s sp",id=tbm,#surname_gen}.
/*
```

```
{lex=claus,phon="",id=cn,#firstname}.
{lex=nielsen,phon="",id=cn,#surname}.
{lex=claus,phon="",id=cn,#firstname_gen}.
{lex=nielsens,phon="",id=cn,#surname_gen}.
*/
{lex=jesper,phon="y eh s p er sp",id=jo,#firstname}.
{lex=olsen,phon="ow l s eh n sp",id=jo,#surname}.
{lex=jespers,phon="y eh s p er s sp",id=jo,#firstname_gen}.
{lex=olsens,phon="ow l s eh n s sp",id=jo,#surname_gen}.

{lex=eric,phon="eh r ih k sp",id=ett,#firstname}.
{lex=thiele,phon="t ih l sp",id=ett,#surname}.
{lex=erics,phon="eh r ih k s sp",id=ett,#firstname_gen}.
{lex=thieles,phon="t ih l s sp",id=ett,#surname_gen}.
/*
{lex=ipke,phon="",id=iw,#firstname}.
{lex=wachsmuth,phon="",id=iw,#surname}.
{lex=ipkes,phon="",id=iw,#firstname_gen}.
{lex=wachsmuths,phon="",id=iw,#surname_gen}.
*/
%%macro
#lab={cat=noun,case=no,nb=sing,semtype=place}.

%% Noun: a2-Places
{lex=meeting_room,phon="m iy t ih ng r uw m sp",id=a2_100,#lab}.
%%{lex=laboratory_for_speech_encoding,phon="",id=a2_101,#lab}.
{lex=speech_coding_lab,phon="s p iy ch k ow d ih ng l ax b sp|
s p iy ch k ow d ih ng l ae b sp",id=a2_101,#lab}.
{lex=speech_laboratory,phon="s p iy ch l ae b ax r ax t ax r ih sp|
s p iy ch l ae b ax r ax t r ih sp",id=a2_102,#lab}.
{lex=speech_lab,phon="s p iy ch l ax b sp|s p iy ch l ae b sp",id=a2_102,#lab}.
{lex=dialogue_laboratory,phon="d ay ax l oh g l ax b aa r ax t r ih sp|
d ay ax l oh g l ax b aa r ax t ax r ih sp",id=a2_103,#lab}.
{lex=dialogue_lab,phon="d ay ax l oh g l ax b sp",id=a2_103,#lab}.
{lex=instrument_repair,phon="ih n s t r uh m er n t r eh p ey r sp",id=a2_105,#lab}.
{lex=computer_room,phon="k ax m p y uw t er r uw m sp",id=a2_221,#lab}.
{lex=office,phon="oh f ih s sp",id=office,#lab}.

%% (Ambiguous a2-places)
{lex=laboratory,phon="l ae b ax r ax t ax r ih sp|l ae b ax r ax t r ih sp",
id=a2_101,#lab}.
{lex=laboratory,phon="l ae b ax r ax t ax r ih sp|l ae b ax r ax t r ih sp",
id=a2_102,#lab}.
{lex=laboratory,phon="l ae b ax r ax t ax r ih sp|l ae b ax r ax t r ih sp",
id=a2_103,#lab}.
```

```
%% Noun: Other
{lex=route,phon="r uw t sp",cat=noun,case=no,nb=sing,valence=pp_pp,
arg1=from,arg2=to,semtype=route}.
%% Verbs
{lex=show,phon="sh ow sp",cat=verb,vform=imp,valence=np_np}.
{lex=point,phon="p oy n t sp",cat=verb,vform=imp,valence=pp,arg2=to}.
{lex=is,phon="ih z sp",cat=verb,vform=ind,prs=3,nb=sing,valence=predicate}.

%% Prep
{lex=in,phon="ih n sp",cat=prep}.
{lex=from,phon="f r oh m sp|f r ax m sp",cat=prep}.
{lex=to,phon="t uw sp|t ax sp",cat=prep}.

%% Pron
{lex=me,phon="m iy sp",cat=pron,subtype=person,case=acc,nb=sing,prs=1}.
%%{lex=him,phon="",cat=pron,subtype=person,case=acc,nb=sing,prs=3}.
%%{lex=who,phon="",cat=pron,subtype=interrog,case=nom,prs=3}.
{lex=whose,phon="hh uw z sp",cat=pron,subtype=interrog,case=gen}.
{lex=this,phon=" dh ih s sp",cat=pron,subtype=demonstr,nb=sing,prs=3}.
{lex=that,phon="dh ae t sp",cat=pron,subtype=demonstr,nb=sing,prs=3}.

%% Determiners
{lex=the,phon="dh ax sp|dh iy sp|dh ah sp",cat=det,def=yes}.

%% Adverbials
{lex=where,phon="w ey r sp",cat=adv, type=interrog}.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%                              SYNTACTIC RULES                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%


%% Show me ... point to
s1_imp = {cat=s, stype=imp}
   [ {cat=vp,vform=imp} ].

%% Where is ...
s1_whq = {cat=s, stype=whq}
   [ {cat=adv, type=interrog},
     {cat=verb, valence=predicate},
     {cat=predicate}
```

```
   ].

%% Whose office is ...
s2_whq = {cat=s, stype=whq}
   [ {cat=np, semtype=interrog},
     {cat=verb, valence=predicate},
     {cat=predicate}
   ].

np_interrog = {cat=np,semtype=interrog}
   [ {cat=pron,subtype=interrog},
     {cat=noun,subtype=place}
   ].

%% e.g. show sbdy sth
vp_1 = {cat=vp, prs=$P, nb=$N,vform=$V}
   [ {cat=verb,prs=$P,nb=$N,vform=$V,valence=np_np},
     {cat=np, case=acc},
     {cat=np, case=no}
   ].

%% e.g. point to sth
vp_2 = {cat=vp, prs=$P, nb=$N}
   [ {cat=verb,prs=$P,nb=$N,valence=pp,arg2=$A},
     {cat=pp, argtype=$A}
   ].

pred_1 = {cat=predicate}
   [ {cat=np, case=no} ].

pp_1 = {cat=pp, argtype=$L}
   [ {cat=prep, lex=$L},
     {cat=np}
   ].

np_gen = {cat=np,case=$C,prs=$P,nb=$N,semtype=$S}
   [ {cat=np,case=gen,semtype=person},
     {cat=np,case=$C,prs=$P,nb=$N,semtype=$S}
   ].

%%me, him (I, he, her, his ...)
np_pron = {cat=np,case=$C,prs=P,nb=$N,semtype=person}
   [ {cat=pron,case=$C,prs=$P,nb=$N} ].

%%tom, toms, broendsted, broendsteds
```

```
np_person1 = {cat=np,case=$C, id=$I,semtype=person}
   [ {cat=proper, case=$C, id=$I} ].

%%tom broendsted, tom broendsteds
np_person2 = {cat=np,case=$C,id=$I,semtype=person}
   [ {cat=proper,case=no,id=$I,surname=no},
     {cat=proper,case=$C,id=$I,surname=yes} ].

np_place1 = {cat=np,case=$C,id=$I,def=yes,semtype=place}
   [ {cat=det,lex=the},
     {cat=noun,case=$C,id=$I,semtype=place}
   ].

np_place2 = {cat=np,case=$C,id=$I,def=no,semtype=place}
   [ {cat=noun,case=$C,id=$I,semtype=place} ].

np_route1 = {cat=np,case=$C,def=yes,semtype=route}
   [ {cat=det,lex=the},
     {cat=noun,case=$C,semtype=route,valence=pp_pp,arg1=$A1,arg2=$A2},
          {cat=pp,argtype=$A1},
          {cat=pp,argtype=$A2}
   ].


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%                              SEMANTIC RULES                             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%


%%show me sth
point1 =
(nlp
   (intention
      (instruction(pointing)),
      #LOCATION,
    time($T)
   )
)
/
{cat=s}
[
{cat=vp}
[
```

```
   {cat=verb,lex=show},
   {cat=np},
   {cat=np,semtype=place}#LOCATION
]
].


%%point to sth
point1 =
(nlp
   (intention
      (instruction(pointing)),
    #LOCATION,
    time($T)
   )
)
/
{cat=s}
[
{cat=vp}
[
   {cat=verb,lex=point},
   {cat=pp}
   [
      {cat=prep,lex=to},
      {cat=np}#LOCATION
   ]
]
].


%%where is <names> office
query1 =
(nlp
   (intention
      (query(where)),
      #LOCATION,
    time($T)
   )
)
/
{cat=s,stype=whq}
[
   {cat=adv,lex=where},
   {cat=verb},
```

```
   {cat=predicate}
   [
      {cat=np}   #LOCATION
   ]
].

%% whose office is this

query2 =
(nlp
   (intention
      (query(who)),
                location(this($Deixis),type(office)),
    time($T)
   )
)
/
{cat=s,stype=whq}
[
   {cat=np}
   [
      {cat=pron,lex=whose},
      {cat=noun,lex=office}
   ],
   {cat=verb,lex=is},
   {cat=predicate}
   [
      {cat=np}
      [
         {cat=pron}
      ]
   ]
].


%%show me the route from <location> to <location>

show_route1 =
(nlp
   (intention
      (instruction(show_route)),
         source(#LOCATION1),
    destination(#LOCATION2),
    time($T)
   )
```

```
)
/
{cat=s}
[
{cat=vp}
[
   {cat=verb,lex=show},
   {cat=np},
   {cat=np}
   [
      {cat=det},
      {cat=noun},
      {cat=pp}
      [
         {cat=prep},
         {cat=np} #LOCATION1
      ],
      {cat=pp}
      [
         {cat=prep},
         {cat=np} #LOCATION2
      ]
   ]
]
].

names_office1 =
(location
   (person($I),type(office))
)
/
{cat=np}
[
   {cat=np,semtype=person,id=$I},
   {cat=np}
].

place1 =
(location
   (place($I))
)
/
{cat=np,semtype=place,id=$I}
.
```

Appendix G

The People

Tom Brøndsted

Tom Brøndsted, www.cpk.auc.dk/~tb, has an M.Sc. in Germanic Philology & General Linguistics from Odense University, Denmark, 1984. In 1984 he was Instructor in phonetics, Dept. of Germanic Philology, in epistemology, Dept. of General Linguistics, Odense University, in 1985 Lecturer at Dept. of Germanic Philology, University of Basel, Switzerland, and Dept. of Germanic Philology, University of Zurich, Switzerland, and in 1991 research computational linguist, Center for PersonKommunikation, Aalborg University, Denmark. He has been involved with organising a number of international meetings including ASDS-87 in Freiburg, Germany, IASS-89 in Zurich, Switzerland and DALF-96 at Aalborg, Denmark. He has been president of the Danish Society for Computational Linguistics since 1995. His research interests include Automatic Speech Understanding (ASR) and Natural Language Processing (NLP): Phonemics and speech recognition, prosody (intonation, speech rate), language modelling, syntactic and semantic parsing, generation. His current projects include IntelliMedia 2000+ (http://www.kom.auc.dk/CPK/MMUI), and Reward (http://www.kom.auc.dk/CPK/Speech/Reward).

Paul Dalsgaard

Paul Dalsgaard obtained his degree in Electronic Engineering in 1962 from the Technical University of Denmark. He was employed at the Academy of Electronic Engineering in Copenhagen and Aalborg until 1974, where he joined the then founded Aalborg University. He is a Professor in speech communications at the Center for PersonKommunikation, where his group is involved in Spoken Dialogue Systems, Speech Recognition and Understanding, Speech Analysis and Synthesis and Design of MultiMedia/MultiModal Human-Computer Interfaces. The speech communications group has been involved in a large number of EU-founded projects over the last ten years, which has led to the establishment of demonstrators applying results from ongoing research. Paul Dalsgaard's special interests lie in the area of applying acoustic-phonetic interdisciplinary research in speech processing and

in modelling similar acoustic sounds across languages in an attempt to generalise speech processing multilingually. He is a Senior Member of IEEE, is a board member of the European Speech Communication Association (ESCA) and has had a visiting university position in Canada.

Lars Bo Larsen

Lars Bo Larsen was born in 1959 near Viborg in the Jutland peninsula of Denmark. He attended Aalborg University where he obtained his Master's degree in Electronic Engineering in 1984, specialising in Control Engineering. From 1984 to 1986 he served at Aalborg university as an Assistant Professor, teaching control engineering and joined a research project concerning real-time expert systems for operator communication in powerplants. In 1987 he joined the Center for PersonKommunikation where he worked in the area of expert systems for speech recognition. He later focussed on Spoken Dialogue Systems and more recently also on MultiModal Human Computer Interaction (HCI). He is currently holding a position as Associate Research Professor and is coordinator of the CPKs Spoken Dialogue group and the Master's education in Intelligent MultiMedia at Aalborg University.

Michael Manthey

Michael Manthey, an Associate Professor of computer science, received a B.S. in Mathematics from Renesselaer Polytechnic Institute, USA and M.Sc. and Ph.D. in Computer Science from the State University of New York at Buffalo, USA. Prior to his current position in Aalborg, he taught at Aarhus University, Denmark and the University of New Mexico, Albuquerque, New Mexico, USA. Dr. Manthey's central interest is distributed systems, in particular the description and modelling of self-organizing and distributed natural and artificial systems.

Paul Mc Kevitt

Paul Mc Kevitt is 34 and from Dún Na nGall (Donegal), Ireland on the Northwest of the EU. He is a Visiting Professor of Intelligent MultiMedia Computing at Aalborg University, Denmark and a British EPSRC (Engineering and Physical Sciences Research Council) Advanced Fellow in the Department of Computer Science at the University of Sheffield, Sheffield, England. The Fellowship, commenced in 1994, and releases him from his Associate Professorship (tenured Lecturership) for 5 years to conduct full-time research on the integration of natural language, speech and vision processing. He is currently pursuing a Master's degree in Education at the University of Sheffield. He completed his Ph.D. in Computer Science at the University of Exeter, England in 1991. His Master's degree in Computer Science was obtained from New Mexico State University, New Mexico, US

in 1988 and his Bachelor's degree in Computer Science from University College Dublin (UCD), Dublin, Ireland in 1985. His primary research interests are in Natural Language Processing (NLP) including the processing of pragmatics, beliefs and intentions in dialogue. He is also interested in Philosophy, MultiMedia and the general area of Artificial Intelligence.

Thomas Baltzer Moeslund

Thomas Baltzer Moeslund is 27 and from Randers, Denmark. He is currently a Ph.D. candidate at the Laboratory of Image Analysis (LIA), Aalborg University, Denmark within the area of computer vision and virtual reality interfaces. He has worked as a Research Assistant at the Center for PersonKommunikation (CPK), Aalborg University, Denmark on Intelligent MultiMedia and computer vision. He obtained his Master's degree in Electrical Engineering with a specialization in computer vision from Aalborg University in 1996. During his study he visited computer vision laboratories at the University of Reading, England, University of Tennessee, Knoxville, US and University of California, San Diego, US. His primary research interests are computer vision, intelligent interfaces and MultiModal systems.

Kristian G. Olesen

Kristian G. Olesen is 40 and from the northern Jutland at the top of Denmark. He is Associate Professor at the Department for Medical Informatics and Image Analysis at Aalborg University. He holds a Bachelor's degree in Mathematics and a Master's degree in Computer Science, both obtained at Aalborg University, where he also received his Ph.D. degree in 1992. His research interest is centered around decision support systems, including both methodological issues and development of applications, primarily within the medical domain. The focus of his research is on operationalisation of theories and methods, and he is cofounder and vicechairman of the company HUGIN Expert Ltd. which markets software for Bayesian Networks.

References

Aho, Alfred V., John E. Hopcroft and Jeffrey D. Ullman (1983) *Data structures and algorithms*. Reading, MA: Addison-Wesley.

André, Elisabeth, G. Herzog, and T. Rist (1988) On the simultaneous interpretation of real-world image sequences and their natural language description: the system SOCCER. In *Proceedings of the 8th European Conference on Artificial Intelligence*, 449-454, Munich, Germany.

André, Elisabeth and Thomas Rist (1993) The design of illustrated documents as a planning task. In *Intelligent multimedia interfaces*, M. Maybury (Ed.), 75-93. Menlo Park, CA: AAAI Press.

Bækgaard, Anders (1996) Dialogue management in a generic dialogue system. In *Proceedings of the Eleventh Twente Workshop on Language Technology (TWLT), Dialogue Management in Natural Language Systems*, 123-132, Twente, The Netherlands.

Bækgaard, Anders, T. Brøndsted, A. Roman, and P. Wetzel (1992) *Advanced dialogue design.* SUNSTAR Deliverable IV.6-1. Esprit Project 1094, Center for PersonKommunikation, Aalborg University, Denmark.

Bækgaard, Anders, O. Bernsen, T. Brøndsted, P. Dalsgaard, H. Dybkjær, L. Dybkjær, J. Kristensen, L.B. Larsen, B. Lindberg, B. Maegaard, B. Music, L. Offersgaard, and C. Povlsen (1995) The Danish spoken language dialogue project - a general overview. In *Proceedings of ESCA WS on Spoken Dialogue Systems: theory and applications, Vigsø, Denmark*, Paul Dalsgaard, Lars Bo Larsen, Louis Bovis, and Ib Thomsen (Eds.), 89-92. Center for PersonKommunikation, Aalborg University, Denmark.

Bai, Bo Nygaard, Tom Brøndsted, and Paul Dalsgaard (1998) *Technical specification: service creation tool.* Technical Report, Reward WP3, Deliveable LE-2632-D3.2 (in confidence), Center for PersonKommunikation, Aalborg University, Denmark, March.

Bakman, Lau, Mads Blidegn, Thomas Dorf Nielsen, and Susana Carrasco Gonzalez (1997a) *NIVICO - Natural Interface for VIdeo COnferencing.* Project Report (8th Semester), Department of Communication Technology, Institute 8, Aalborg University, Denmark.

Bakman, Lau, Mads Blidegn, and Martin Wittrup (1997b) *Improving human computer interaction by adding speech, gaze, tracking and agents to a WIMP based environment.* Project Report (9th/10th Semester), Department of Communication Technology, Institute 8, Aalborg University, Denmark.

Bech, A. (1991) Description of the EUROTRA framework. In *The Eurotra Formal*

Specifications, Studies in Machine Translation and Natural Language Processing, C. Copeland, J. Durand, S. Krauwer, and B. Maegaard (Eds), Vol. 2, 7-40. Luxembourg: Office for Official Publications of the Commission of the European Community.

Bresnan, J. (Ed.) (1982) *The mental representation of grammatical relations.* Cambridge, Mass.: MIT Press.

Bruegge, Bernd and Ben Bennington (1996) Applications of wireless research to real industrial problems: applications of mobile computing and communication. In *IEEE Personal Communications*, 64-71, February.

Brøndsted, T. (1998) *nlparser.* WWW: http://www.kom.auc.dk/~tb/nlparser.

Brøndsted, T., and L.B. Larsen (1994) *Representation of acoustic and linguistic knowledge in continuous speech recognition.* Spoken Language Dialogue Systems, Technical Report 5, Center for PersonKommunikation, Aalborg University, Denmark.

Carenini, G., F. Pianesi, M. Ponzi and O. Stock (1992) *Natural language generation and hypertext access.* IRST Technical Report 9201-06, Instituto Per La Scientifica E Tecnologica, Loc. Pant e Di Povo, I-138100 Trento, Italy.

Chomsky, N. (1956) Three models for the description of language. In *I.R.E. Transactions on Information Theory*, Vol. IT-2, 113-124, September.

Chomsky, N. (1957) *Syntactic structures.* The Hague, The Netherlands: Mouton.

Christensen, Heidi, Børge Lindberg and Pall Steingrimsson (1998a) *Functional specification of the CPK Spoken LANGuage recognition research system (SLANG).* Technical Report, Center for PersonKommunikation, Aalborg University, Denmark, March.

Christensen, Heidi, Børge Lindberg and Pall Steingrimsson (1998b) *Technical specification of the CPK Spoken LANGuage recognition research system (SLANG).* Technical Report, Center for PersonKommunikation, Aalborg University, Denmark, March.

CPK Annual Report (1998) *CPK Annual Report.* Center for PersonKommunikation (CPK), Fredrik Bajers Vej 7-A2, Institute of Electronic Systems (IES), Aalborg University, DK-9220, Aalborg, Denmark.

Dalsgaard, Paul and A. Baekgaard (1994) Spoken Language Dialogue Systems. In *Prospects and Perspectives in Speech Technology: Proceedings in Artificial Intelligence*, Chr. Freksa (Ed.), 178-191, September. München, Germany: Infix.

Denis, M. and M. Carfantan (Eds.) (1993) *Images et langages: multimodalité et modelisation cognitive.* Actes du Colloque Interdisciplinaire du Comité National de la Recherche Scientifique, Salle des Conférences, Siège du CNRS, Paris, April.

Dennett, Daniel (1991) *Consciousness explained.* Harmondsworth: Penguin.

Earley, Jay (1970) An efficient context-free parsing algorithm. In *Communications of the ACM*, Vol 13, 2, February.

Engberg, Inger Samsø, Anya Varnich Hansen, Ove Andersen, and Paul Dalsgaard (1997) Design, recording and verification of a Danish emotional speech database. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech '97)*, Vol. 4, 1695-1698, Rhodos, Greece, September.

Fink, G.A., N. Jungclaus, H. Ritter, and G. Sagerer (1995) A communication framework for heterogeneous distributed pattern analysis. In *Proc. International Conference on*

Algorithms and Applications for Parallel Processing, V. L. Narasimhan (Ed.), 881-890. IEEE, Brisbane, Australia.

Fink, Gernot A., Nils Jungclaus, Franz Kummert, Helge Ritter and Gerhard Sagerer (1996) A distributed system for integrated speech and image understanding. In *Proceedings of the International Symposium on Artificial Intelligence*, Rogelio Soto (Ed.), 117-126. Cancun, Mexico.

Fraser N. and P. Dalsgaard (1996) Spoken dialogue systems: a European perspective. In *Proceedings of the International Symposium on Spoken Dialogue (ISSD 96), October, Wyndham Franklin Plaza Hotel, Philadelphia, US*, Fujisaki, Hiroya (Ed.), 25-36. Tokyo, Japan: Acoustical Society of Japan (ASJ).

Garofolo, J.S. (1988) *The structure and format of the Darpa TIMIT CD-ROM prototype.* Darpa Timit CD-ROM, an ccoustic phonetic continuous speech database, TT, MIT, US NIST, SRI December 1988, pp. 1-9. (distributed on the DARPA TIMIT CD-ROM).

Garofolo, John S., Lori F. Larnel, William F. Fisher, Jonathon G. Fiscus, David S. Pallett, and Nancy L. Dahlgren (1993) *DARPA TIMIT acoustic-phonetic continuous speech corpus.* US NIST Internal Report 4930. NTIS, U.S. Department of Commerce, Port Royal Road, Springfield, VA.

Gazdar, G., E. Klein, G.K. Pullum, and I.A. Sag (1985) *Generalized phrase structure grammar.* Oxford, England: Blackwell Publishing and Cambridge, Massachusetts, US: Harvard University Press.

Gonzalez, R.C. and R.E. Woods (1993) *Digital image processing.* Reading, Massachusetts: Addison Wesley.

Herzog, G. and G. Retz-Schmidt (1990) Das System SOCCER: Simultane Interpretation und naturalichsprachliche Beschreibung zeitveranderlicher Szenen. In *Sport und Informatik*, J. Perl (Ed.), 95-119. Schorndorf: Hofmann.

Herzog, G., C.-K. Sung, E. Andre, W. Enkelmann, H.-H. Nagel, T. Rist, and W. Wahlster (1989) Incremental natural language description of dynamic imagery. In *Wissenbasierte Systeme. 3. Internationaler GI-Kongress*, C. Freksa and W. Brauer (Eds.), 153-162. Berlin: Springer-Verlag.

IEEE Spectrum (1996) *Technology 1996: analysis and forecast issue Can we talk? the great hookup debate.* January.

IEEE Spectrum (1997) *Sharing virtual worlds: avatars, agents, and social computing.* Vol. 34, No. 3, March.

Infovox (1994) *INFOVOX: Text-to-speech converter user's manual (version 3.4).* Solna, Sweden: Telia Promotor Infovox AB.

JAI (1996) *Cameras for surveillance.* Copenhagen, Denmark: JAI A-S.

Jensen, Finn V. (1996) *An introduction to Bayesian Networks.* London, England: UCL Press.

Jensen, Frank (1996) Bayesian belief network technology and the HUGIN system. In *Proceedings of UNICOM seminar on Intelligent Data Management*, Alex Gammerman (Ed.), 240-248. Chelsea Village, London, England, April.

Jensen, Jesper, Claus Nielsen, Ove Andersen, Egon Hansen, and Niels-Jørn Dyhr (1998) A speech synthesizer with modelling of the Danish "stød". In *Proceedings of the IEEE Nordic Signal Processing Symposium (Norsig '98)*, 8-11, June.

Kay, M. (1985) Parsing in functional unification grammar. In *Natural Language Parsing*, D.R. Dowty, Lauri Karttunen and Arnold M. Zwicky (Eds.), 251-278. Cambridge University Press: Cambridge, Engand.

Kjærsdam, Finn and Stig Enemark (1994) *The Aalborg experiment - project innovation in university education.* Aalborg, Denmark: Aalborg Universitetsforlag (Aalborg University Press).

Kosslyn, S.M. and J.R. Pomerantz (1977) Imagery, propositions and the form of internal representations. In *Cognitive Psychology*, 9, 52-76.

Larsen, L.B. (1996) Voice controlled home banking - objectives and experiences of the Esprit OVID project. In *Proceedings of IEEE Third Workshop on Interactive Voice Technology for Telecommunications Applications (IVTTA-96)*, 45-48, New Jersey, US, September (IEEE 96-TH-8178).

Lausen, Henrik (1997) *LaserXI(2.2) documentation.* Laser Interface, Center For Advanced Technology (CAT), Roskilde, Denmark.

Lee, K. (1988) *Large-vocabulary speaker-independent continuous speech recognition: The SPHINX System.* Technical Report, CMU-CS-88-148, Carnegie Mellon University, Pittsburgh, PA, April.

Leth-Espensen, Poul and Børge Lindberg (1995) *Application of microphone arrays for remote voice pickup - a feasibility study - final report. RVP (Remote Voice Pickup) - Report 2 (Commercial in confidence).* Center for PersonKommunikation, Aalborg University, Denmark.

Leth-Espensen, Poul and Børge Lindberg (1996) Separation of speech signals using eigenfiltering in a dual beamforming system. In *Proc. IEEE Nordic Signal Processing Symposium (NORSIG)*, Espoo, Finland, September, 235-238.

Lindberg, B., and J. Kristiansen (1995) *Real-time continuous speech recognition within dialogue systems.* Spoken Language Dialogue Systems, Report 8, Center for PersonKommunikation, Aalborg University, Denmark.

Maaβ, Wolfgang, Peter Wizinski, Gerd Herzog (1993) *VITRA GUIDE: Multimodal route descriptions for computer assisted vehicle navigation.* Bereich Nr. 93, Universitat des Saarlandes, FB 14 Informatik IV, Im Stadtwald 15, D-6600, Saarbrucken 11, Germany, February.

Maegaard, B., H.D. Maas (1985) *Definition of the Eurotra user language.* Final Report (ETS-7-DK/D), submitted to the Commission of the European Communities (CEC).

Manthey, Michael J. (1991) *US Patents 4,829,450, 5,367,449, and others pending.* The intent is to license freely (on request) to individuals and research or educational institutions for non-profit use..

Manthey, Michael J. (1994) Toward an information mechanics. In *Proceedings of the Third IEEE Workshop on Physics and Computation*, D. Matzke, (Ed.), 95-109. Dallas, Texas, November.

Manthey, Michael J. (1997a) *Distributed computation, the twisted isomorphism, and autopoiesis.* Technical Report R-97-5007, Department of Computer Science, Aalborg university, Denmark, June.

Manthey, Michael J. (1997b) *The phase web paradigm and anticipatory systems, two short papers.* Technical Report R-97-5006, Department of Computer Science, Aalborg university, Denmark, June.

Manthey, Michael J. (1998a) Distributed Computation, the Twisted Isomorphism, and Auto-Poiesis. In *Proceedings of the First International Conference on Computing Anticipatory Systems (CASYS 97)*, D. Dubois, (Ed.). Liege, Belgium, August.

Manthey, Michael J. (1998b) The Phase Web Paradigm. In *International Journal of General Systems, special issue on General Physical Systems Theories*, K. Bowden (Ed.). in press.

Manthey, Michael J. (1998c) *topsy.* WWW: http://www.cs.auc.dk/topsy/.

Maybury, Mark (1991) Planning multimedia explanations using communicative acts. In *Proceedings of the Ninth American National Conference on Artificial Intelligence (AAAI-91)*, Anaheim, CA, July 14-19.

Maybury, Mark (Ed.) (1993) *Intelligent multimedia interfaces.* Menlo Park, CA: AAAI Press.

Maybury, Mark (Ed.) (1997) *Intelligent multimedia information retrieval.* Menlo Park, CA: AAAI/MIT Press, http://www.aaai.org:80/Press/Books/Maybury-2/.

Maybury, Mark and Wolfgang Wahlster (Eds.) (1998) *Readings in intelligent user interfaces.* Los Altos, CA: Morgan Kaufmann Publishers.

Mc Kevitt, Paul (1994) Visions for language. In *Proceedings of the Workshop on Integration of Natural Language and Vision processing*, Twelfth American National Conference on Artificial Intelligence (AAAI-94), Seattle, Washington, USA, August, 47-57.

Mc Kevitt, Paul (Ed.) (1995/1996) *Integration of Natural Language and Vision Processing (Vols. I-IV).* Dordrecht, The Netherlands: Kluwer-Academic Publishers.

Mc Kevitt, Paul (1997) SuperinformationhighwayS. In *"Sprog og Multimedier" (Speech and Multimedia)*, Tom Brøndsted and Inger Lytje (Eds.), 166-183, April 1997. Aalborg, Denmark: Aalborg Universitetsforlag (Aalborg University Press).

Minsky, Marvin (1975) A framework for representing knowledge. In *The Psychology of Computer Vision*, P.H. Winston (Ed.), 211-217. New York: McGraw-Hill.

Moeslund, Thomas B., Lau Bakman, and Mads Blidegn (1998) *Controlling a moveable laser from a PC.* Technical Report R-98-1002, Center for Personkommunikation (CPK), Institute of Electronic Systems (IES), Aalborg University, Denmark.

Music, B., L. Offersgaard, D. Christensen (1994) *The NLP module.* Spoken Language Dialogue Systems, Report 7, Center for PersonKommunikation, Aalborg University, Denmark.

mySQL (1998) *WWW: http://www.tcx.se/.* Stockholm, Sweden: T.c.X DataKonsult AB.

Neumann, B. and H.-J. Novak (1986) NAOS: Ein System zur naturalichsprachlichen Beschreibung zeitveranderlicher Szenen. In *Informatik. Forschung und Entwicklung*, 1(1): 83-92.

Nielsen, Claus, Jesper Jensen, Ove Andersen, and Egon Hansen (1997) *Speech synthesis based on diphone concatenation.* Technical Report, No. CPK971120-JJe (in confidence), Center for PersonKommunikation, Aalborg University, Denmark.

Nielsen, Jørgen (1997) *Distributed applications communication system applied on IntelliMedia WorkBench.* Project Report (8th Semester), Department of Medical Informatics and Image Analysis (MIBA), Institute 8, Aalborg University, Denmark.

Odell, Julian, Dave Ollason, Valtcho Valtchev, and David Whitehose (1997) *The HAPI Book (version 1.0): a description of the HTK application programming interface.* Cambridge, England: Entropic Cambridge Research Laboratory Ltd..

Okada, Naoyuki (1996) Integrating vision, motion and language through mind. In *Integration of Natural Language and Vision Processing, Volume IV, Recent Advances*, Mc Kevitt, Paul (ed.), 55-80. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Okada, Naoyuki (1997) Integrating vision, motion and language through mind. In *Proceedings of the Eighth Ireland Conference on Artificial Intelligence (AI-97), Volume 1*, 7-16. University of Uster, Magee, Derry, Northern Ireland, September.

Partridge, Derek (1991) *A new guide to Artificial Intelligence.* Norwood, New Jersey: Ablex Publishing Corporation.

Pentland, Alex (Ed.) (1993) *Looking at people: recognition and interpretation of human action.* IJCAI-93 Workshop (W28) at The 13th International Conference on Artificial Intelligence (IJCAI-93), Chambéry, France, EU, August.

Pollard, C. (1984) *Generalized phrase structure grammars, head grammars, and natural languages.* Ph.D. thesis, Department of Linguistics, Stanford University, CA, US.

Pollard, C., and I.A. Sag (1994) *Head-driven phrase structure grammar.* Chicago, IL, US: Univeristy of Chicago Press.

Povlsen, C. (1994) *Sublanguage definition and specification.* Spoken Language Dialogue Systems, Report 4, Center for PersonKommunikation, Aalborg University, Denmark.

Power, Kevin, Caroline Matheson, Dave Ollason and Rachel Morton (1997) *The grapHvite book (version 1.0).* Cambridge, England: Entropic Cambridge Research Laboratory Ltd..

Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling (1990) *Numerical recipes in C - the art of scientific computing.* Cambridge, England: Cambridge University Press.

Pylyshyn, Zenon (1973) What the mind's eye tells the mind's brain: a critique of mental imagery. In *Psychological Bulletin*, 80, 1-24.

Retz-Schmidt, Gudala (1991) Recognizing intentions, interactions, and causes of plan failures. In *User Modelling and User-Adapted Interaction*, 1: 173-202.

Retz-Schmidt, Gudala and Markus Tetzlaff (1991) *Methods for the intentional description of image sequences.* Bereich Nr. 80, Universitat des Saarlandes, FB 14 Informatik IV, Im Stadtwald 15, D-6600, Saarbrucken 11, Germany, EU, August.

Rich, Elaine and Kevin Knight (1991) *Artificial Intelligence.* New York: McGraw-Hill.

Rickheit, Gert and Ipke Wachsmuth (1996) Collaborative Research Centre "Situated Artificial Communicators" at the University of Bielefeld, Germany. In *Integration of*

Natural Language and Vision Processing, Volume IV, Recent Advances, Mc Kevitt, Paul (ed.), 11-16. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Roehle, B. (1997) Channeling the data flow. In *IEEE Spectrum*, March, 32-38.

Ross, S.M. (1987) *Introduction to probability and statistics for engineers and scientists.* Chichester, England: John Wiley and Sons.

Rudnicky, Alexander I., Stephen D. Reed, Eric H. Thayer (1996) SpeechWear: a mobile speech system. In *Proceedings of International Symposium on Spoken Dialogue (ISSD 96), October 2-3, Wyndham Franklin Plaza Hotel, Philadelphia, USA*, Fujisaki, Hiroya (Ed.), 161-164. Tokyo, Japan: Acoustical Society of Japan (ASJ).

Shieber, S.M. (1984) The design of a computer language for linguistic information. In *Proceedings of the 10th International Conference on Computational Linguistics (Coling-84)*, Stanford, CA, 362-366.

Shieber, S.M. (1985) Using restriction to extend parsing algorithms for complex-feature-based formalims. In *Proceedings of the 22nd Annual Meeting of the Association for Computational Linguistics*, Chicago, Il, 145-152.

Shieber, S.M. (1986) *An introduction to unification based approaches to grammar.* CSLI Lecture Notes Series, Number 4. Stanford, CA: Center for Study of Language and Information

Smailagic, Asim and P. Siewiorek (1996) Matching interface design with user tasks: modalities of interaction with CMU wearable computers. In *IEEE Personal Communications*, 14-25, February.

Stock, Oliviero (1991) Natural language and exploration of an information space: the AL-Fresco Interactive system. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, 972-978, Darling Harbour, Sydney, Australia, August.

Thórinsson, Kris R. (1996) *Communicative humanoids: a computational model of psychosocial dialogue skills.* Ph.D. thesis, Massachusetts Institute of Technology.

Thórinsson, Kris R. (1997) Layered action control in communicative humanoids. In *Proceedings of Computer Graphics Europe '97*, June 5-7, Geneva, Switzerland.

Tsai, R.Y. (1987) A versatile camera calibration for high-accuracy 3D machine vision metrology using off-the self TV cameras and lenses. In *IEEE Journal of Robotics and Automation*, Vol. 3, No. 4, August.

Tuns, Nicolae G. and Thomas Dorf Nielsen (1998) *Experimenting with phase web as AI support in the CHAMELEON system.* Project Report (9th semester), Department of Computer Science, Institute 8, Aalborg University, Denmark.

Uhlin, Tomas and Johansson, Kenneth (1996) *Autonomous mobile systems: a study of current research.* The Royal Institute of Technology (KTH), Stockholm, Sweden.

von Hahn, Walther (1997) *Personal communication.* .

Wahlster, Wolfgang (1988) *One word says more than a thousand pictures: On the automatic verbalization of the results of image sequence analysis.* Bereich Nr. 25, Universitat des Saarlandes, FB 14 Informatik IV, Im Stadtwald 15, D-6600, Saarbrucken 11, Germany, February.

Wahlster, Wolfgang, Elisabeth Andrè, Wolfgang Finkler, Hans-Jurgen Profitlich, and Thomas Rist (1993) Plan-based integration of natural language and graphics generation. In *Artificial Intelligence, Special issue on natural language generation*, 63, 387-427.

Waibel, Alex, Minh Tue Vo, Paul Duchnowski and Stefan Manke (1996) Multimodal interfaces. In *Integration of Natural Language and Vision Processing, Volume IV, Recent Advances*, Mc Kevitt, Paul (Ed.), 145-165. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Waltz, David (1975) Understanding line drawings of scenes with shadows. In *The psychology of computer vision*, Winston, P.H. (Ed.), 19-91. New York: McGraw-Hill.

Winograd, T. (1983) *Language as a cognitive process - Volume I: Syntax*. Reading, Massachusetts: Addison-Wesley.

Young Steve J., N.H. Russell, and J.H.S. Thornton (1989) *Token passing: a simple conceptual model for connected speech recognition systems*. Technical Report, Cambridge University Engineering Department, Cambridge, England.

Young, Steve J., Joop Jansen, Julian Odell, Dave Ollason, and Phil Woodland (1996) *The HTK book (version 2.0)*. Cambridge, England: Entropic Cambridge Research Laboratory Ltd..