SUPERPOWER: Towards a fully integrated model of planning, action and conversation.

CHARLES G. BUTTON, B.Sc. (Birmingham), M.Sc. (Warwick)

Thesis submitted for the Degree of Doctor of Philosophy, in the Department of Psychology, Faculty of Science, University of Glasgow, June 1992.

Thesis
9285
Copy 1

Contents

Figures

Acknowledgements.

Keith Oatley assisted with the empirical investigation and corrected numerous earlier drafts of this work described in chapter 2.

Simon Garrod offered many helpful comments during the re-drafting stages. B. Neill and T. Kennedy also assisted with some proof reading.

Technical assistance in terms of computing facilities within the Psychology department was also gratefully appreciated. This work was further supported by a three year SERC grant from Sept 1987 - Sept 1990.

Finally Steve Draper's assistance over the last five years could not possibly be measured or explained, but amongst many things, acted as my supervisor and provided the main ideas behind chapters 2, 4, 5 and 6 and constantly supplied me with ideas about the parallels between conversation and action.

Abstract:

Following Austin's work, some theories have attempted to develop the idea that utterances can be understood as actions. In Artificial Intelligence there has been a body of work trying to do this in terms of joint planning. Previous work has been of two kinds: building up from the basic units of utterances and speech acts, or building down from external goals and trying to model connected sequences of utterances i.e. conversation. This thesis extends the latter tradition.

After an initial empirical investigation, two new claims are made. Firstly the computer program, listed in the appendix, shows how time slicing changes dialogue outcome in interesting ways. Secondly an exhaustive planning framework for Power's robot world provides agents with better conversational skills than those reported so far.

Finally the thesis discusses the drawbacks encountered in the empirical investigation and proposes a new basic unit, NEOTELL, a single generic unit of mutual agreement out of which whole conversations can be formed and an agents external goals achieved.

Chapter 1.

Introduction.

1.0 Introduction.

The following thesis stems from the insight due to Austin (1962,1970) that utterances are actions and not propositions. Within Artificial Intelligence this has led to implementing, Power (1979), a theory of joint planning on a computer in such a way that it strives to understand how action sequences are strung together. Finally it should also be embodied within a theory of purposeful communication as whole.

Figure 1.1 represents four examples of interaction between two robots/humans. On the left side is a conversational exchange and on the right side the couple interact by touch only. In the first pair of exchanges the

1. John: Mary

2. Mary: Yes

OR

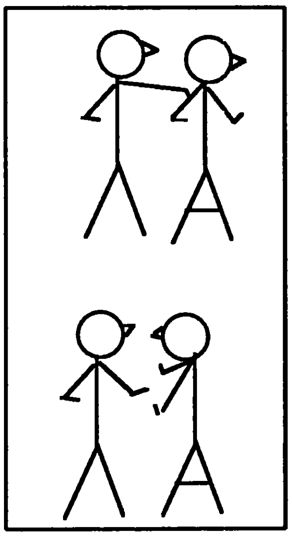


Figure 1.1 The equivalent action for a call by name is a tap on the shoulder.

initiation involves a "summons": either John calls Mary (conversational) or he taps her on the shoulder (non-conversational). Similarly in the second pair Mary responds with a "Yes" (conversational) or by nodding her head (non-conversational). In Figure 1.2 both robots/humans initiate a summons either by "a call by name" or a "tap on the shoulder" simultaneously and I am concerned with the problem of how agents coordinate such communications.

1. John. Mary ... 2. Mary. John **OR**

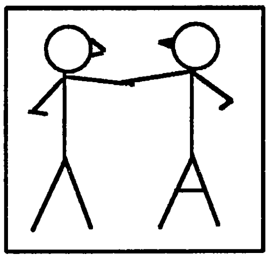


Figure 1.2 Agents Mary and John speak at the same time.

These are crystal clear examples of how communication can occur through either language or action. Since they are the simplest I shall be studying them rigorously until a theory evolves that provides the foundation necessary to build a complete theory of conversation.

Although throughout the thesis I shall be in the main considering robot dialogue simulation. I am interested in

Figure 1.3 Human purposeful dialogue.

human purposeful dialogue similar to that illustrated in figure 1.3 above. I shall be trying to compare this dialogue with the dialogue generated using a computer simulation.

I am not interested in purposeful monologue as

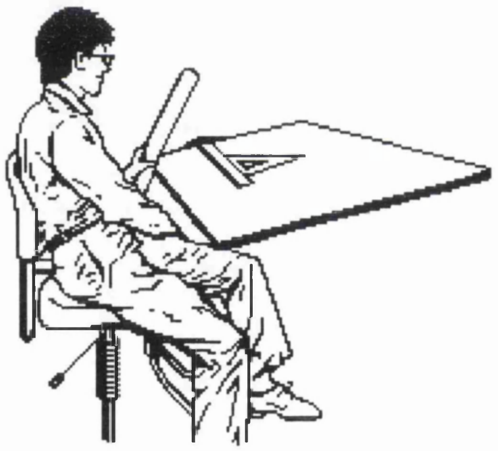


Figure 1.4 Human purposeful monologue.

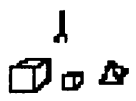
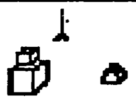
illustrated in figure 1.4 above. Nor one-sided dialogue as illustrated in Weizenbaum's (1966) simulation below.

1. Human:	Men are all alike.
2. Computer:	In what way?
3. Human:	They're always bugging us about something or other.
4. Computer:	Can you think of a specific example?
5. Human:	Well, my boyfriend made me come here.
6. Computer:	Your boyfriend made you come here.?
7. Human:	He says I'm depressed much of the time.
8. Computer:	I am sorry to hear you are depressed.
9. Human:	It's true. I'm unhappy.
10. Computer:	Do you think coming here will help you not to be unhappy?
11. Human:	I need some help, that much seems certain.
12. Computer:	What would it mean to you if you got some help?

.
.
.

<u>Figure 1.5 ELIZA, Weizenbaum (1966). Connectedness in terms of minimal understanding.</u>

Similarly, I am not going to be concerned with sub-dialogues as illustrated with the Winograd (1972) SHRDLU program below.

1  Put the small block on the big block. >Okay.	2  Is the big block under the small block? >Yes.	3 Put the pyramid on the big block. >Okay.
4 Did you touch any block before you put the pyramid on the big block? >Yes. The small one.	5 Can the table pick up blocks? >No. Put the pyramid on the table. >Okay.	6

<u>Figure 1.6 Connectedness, Winograd (1972). Connectedness in terms of action.</u>

Finally, before we go any further I should also make it

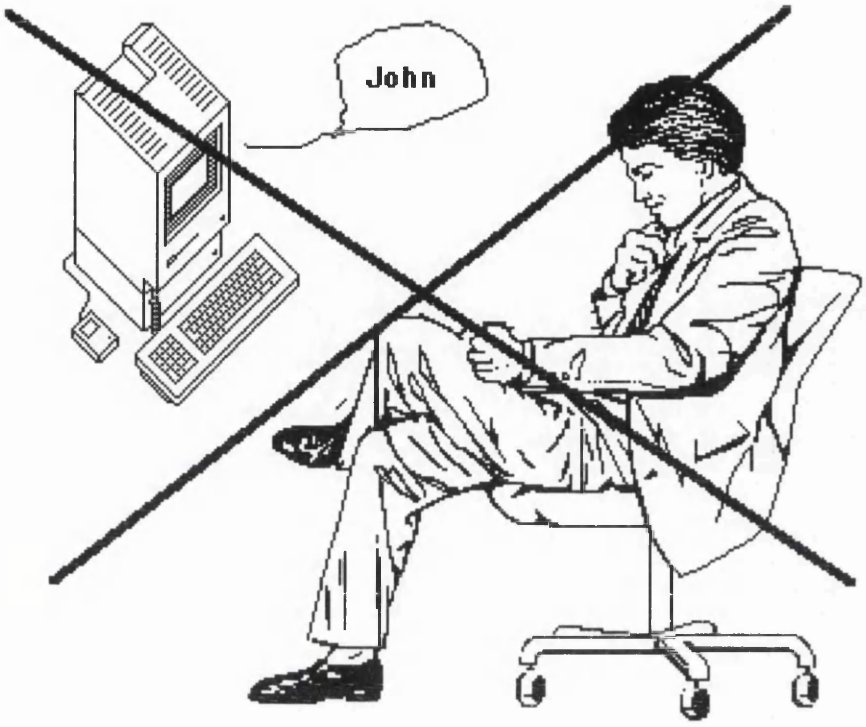


Figure 1.7 Natural language generation.

quite clear that we are not going to be concerned with the above dialogue illustrated in figure 1.7.

1.1 Communication not language.

The first question I wish to consider is that communication can be expressed either through language or action and therefore a dialogue system must have both of these methods at its disposal. Without this feature the dialogue becomes disjoint[1] (see also chapter 3 for an in depth discussion about other systems). Next is the point that

[1] That is to say the program has two different reasoning processes for similar problems. For example, in Power (1979), looking to see whether the bolt is up can be reasoned either within the physical planning algorithm or within a conversational exchange. There is nothing wrong with this except that in some instances (see also chapter 6) it makes it harder to provide a standard theory.

conversational coordination is a feature of dialogue as opposed to monologue. Thus I shall be concerned with the problem of whether or not the conversation succeeds as a whole rather than just in terms of its individual component utterances.

There is also the controversy of efficiency (saying just enough to communicate the purpose, e.g. "John, open the door now.") versus explicitness (saying everything that can be said e.g. "John, Could you please open the door when it is most convenient for you.") that was first discussed by Searle (1969,1975) and Grice (1975) and more recently by (Power 1987, Stubbs 1986). In linguistic philosophy, this was discussed in terms of direct and indirect speech acts and the principle of expressibility. Natural conversation was considered as fulfilling the right balance between what is said and what can be assumed by following Grice's maxim of quantity (see Stubbs 1986: p9).

I shall be adopting explicitness which encourages the idea that a computer model of dialogue must be able to say all that can be said before it is concerned with efficiency. Other systems treatment of this are dealt within the discussion of existing literature in chapter 3.

Then there is the question of coherence. How do utterances hang together? Hirst (1981) studied the problem in relation to anaphoric reference. Garrod and Anderson (1987)

1. Human:	What does the box contain?
2. Computer:	The blue pyramid and the blue block.
3. Human:	What is the pyramid supported by?
4. Computer:	The box.
5. Human:	How many blocks are not in the box?
6. Computer:	Six of them.
7. Human:	Can the table pick up blocks?
8. Computer:	No.
9. Human:	Can a pyramid support a pyramid?
10. Computer:	I don't know.
11. Human:	Will you please stack both of the red blocks and either a green cube or a pyramid?
12. Computer:	Okay.
13. Human:	Had you touched any pyramid before you put the green one on the little cube?
14. Computer:	Yes, the green one.
15. Human:	When did you pick it up?
16. Computer:	While I was stacking up the red cube, a large red block and a large green cube.

Figure 1.8 SHRDLU, Winograd (1972). Connectedness in terms of pronoun reference.

studied this problem in terms of semantic coordination and Winograd (1972), see figure 1.8, discussed this in terms of pronoun reference. In this thesis I shall be studying coherence by considering conversation as though it were coherent action. The first stages of this involve considering alternative communicative methods for achieving the same goal. On the one hand simple coordinated action sequences and on the other by action sequences that are coordinated through discussion.

This work then follows on from Power (1979), who modelled a kind of conversation that made use of a physical setting. In his system, the conversational and physical mechanisms were, in the main, detached from one another and initially I shall be concerned with how to improve the coordination of both these processes. As a starting point, I shall consider the programs of Power (1979) and Houghton (1986). I shall make no contribution to "Artificial Intelligence Joint planning" nor any contribution to a kind of "planning algorithm" but instead I shall try to apply Artificial Intelligence planning to the development of a theory about conversational skills.

This thesis will be concerning itself with the problem of the coordination of semantically planned sentences as opposed to problems of syntax or natural dialogue generation. Finally I shall be concerned with the problem of what constitutes a unit in conversation. A number of definitions have been offered so far: Power in terms of a conversational procedure and Houghton in terms of an Interactional frame. These replaced older attempts such as the adjacency pair, Sacks, Schegloff and Jefferson (1974), and the speech act, Searle (1969), definitions.

1.2 Summary story.

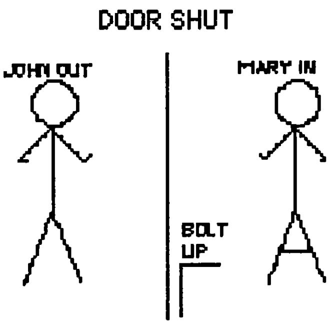
The first point to note is that there are physical and conversational equivalents that allow us to express the same thing (see the implementation discussion in chapter 5 and the design chapter 6). For example, in figure 1.9, if you want to

```
                  (State of the world is
                  [door shut, bolt up, John out, Mary in])

1.  John:         Will you help me get in?
2.  Mary:         By all means.
3.  John:         I suggest we get the door open and then I move.
4.  Mary:         All right.
5.  Mary:         How do you get the door open?
6.  John:         I don't know.
7.  Mary:         Shall we do an experiment?
8.  John:         Okay.
9.  Mary:         I suggest I push the door.
10. John:         All right.

                  (State of the world is
                  [door open, bolt up, John out, Mary in])

11. Mary:         I have pushed the door.
12. John:         I see.
13. Mary:         The door has changed position.
14. John:         I see.
15. Mary:         The door is now open.
16. John:         Right.
17. Mary:         Pushing the door causes the door to change position,
                  when you are in.
18. John:         I see.
```

Figure 1.9 Simplified dialogue of SUPERPOWER (see program listing given in the appendix) illustrating experimental joint planning in which exhausting all permutations by trial and error allows the goal to be achieved.

know how to do something you can either ask for advice or experiment until you find the answer.

The next problem to address is how agents assess the new situation just after an action has occurred (see the dialogue simulation of time slicing discussed in chapter 4 and the section on Unexpected planning in chapter 5). For example, in Power (1979) fig 1.10, this is embedded within the



(State of the world is now
[John out, Mary in, bolt up, door shut])

1. John:	Mary
2. Mary:	Yes.
3. John:	I want to suggest a goal.
4. Mary:	Go ahead.
5. John:	Will you help me get in?
6. Mary:	By all means.
7. John:	I suggest we get the door open and then I move.
8. Mary:	All right.
9. John:	I suggest that you push the door.
10 Mary:	All right.

(State of the world is now
[door open, bolt up, John out, Mary in])

11. Mary:	I have pushed the door.
12. John:	I see.
13. Mary:	The door has changed position.
14. John:	Yes.
15. Mary:	The door is now open.
16. John:	Right.

(State of the world is now
[door open, bolt up, John in, Mary in])

17. John:	I have moved.
18. Mary:	I see.
19. John:	I have changed position.
20. Mary:	Yes.
21. John:	I am now in.
22. Mary:	Right.

Figure 1.10 Power's robot world. Simplified dialogue illustrating classical joint planning techniques for achieving a single physical goal, without utterance priming, adapted from Power (1979).

conversational procedure. In a reactive environment they must also coordinate what they learn. Next is the problem of how one robot responds to the surprises that are thrown on it. The problem of reacting and responding to what actually happened versus what to expect. What counts is the combination of strong expectation versus surprises. The time slicing (see the Monte Carlo dialogue simulation discussed in chapter 4) illustration, figure 1.11 below, demonstrates robustness for conversation. This involves the problem of coordinating a discussion in which both agents are allowed to think simultaneously. As a first step a definition of an increment of program code is required, since at any stage a conversational

	Dialogue A Input variation: John Slow Mary Fast		Dialogue B Input variation: John Fast Mary slow
1. John:	Mary.	1. John:	Mary.
2. Mary:	Yes.	2. Mary:	Yes.
5. John:	Will you help me get the door open?	5. John:	Will you help me get the door open?
6. Mary:	By all means.	6. Mary:	By all means.
9. Mary:	I suggest that I push the door.	13. John:	Can you push the door?
10. John:	Mary.	14. Mary:	Yes.
11. Mary:	Yes.		
		19. John:	Is the bolt up?
14. John:	Is the bolt up?	20. Mary:	Yes.
15. Mary:	Yes.		
		21. John:	I suggest that you push the door.
16. John:	All right.	22. Mary:	All right.
	(State of the world is now [John out, Mary in, bolt up, door open])		(State of the world is now [John out, Mary in, bolt up, door open])

Figure 1.11 Time slicing dialogue from a run of the program SUPERPOWER, listed in the appendix, illustrating a feature of continuity (i.e. the relative speeds of thought of the robots) that can be explained in terms of the program's dynamic (i.e the alternative dialogues A and B) and static (i.e. the listing of the program SUPERPOWER in the appendix) structural representations.

program needs to be able to critically reason about itself. Next I have extended the way in which robots can use knowledge through the use of testing (see section on Instrumental planning discussed in chapter 5). Thus if an agent and partner do not know something they can act out a test in order to infer what happens as a consequence of such a test[2] .

Finally, there is the general issue about incorporating different levels of goals that are brought to the conversation (see the empirical investigation discussed in chapter 2). In brief, this theory involves defining four levels. At the top level 1 are the goals external to the conversation, at the next level 2 are the topics or sub-goals that are achievable through conversation, at level 3 is the unit of a coordinated exchange of utterances and finally at level 4 is the coordination of the mode of communication. In order to explore this notion I have

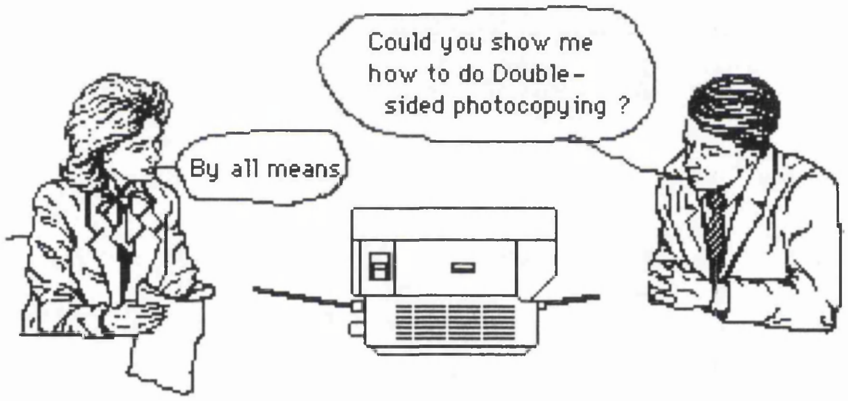


Figure 1.12. The double-sided photocopying task (conversation by two humans).

incorporated an experiment to see whether subjects could be trained to adhere to this theory of levels. In doing so I have incorporated two tasks (illustrated in figures 1.12 and 1.13) that require cooperation in order to achieve the main goal. The

[2] In answer to a direct question such as "Is the bolt up?" there are four main alternatives (1) Yes (2) No (3) I don't know (as mentioned here and discussed in detail in chapter 5) (4) A meta-action response such as pushing the door (section 6.3.7).

conclusions about these cooperative tasks also remind us of the difference between natural conversation and those simulated by programs. In particular some of the data could be interpreted as throwing doubt as to whether human communications coordinate their activity in

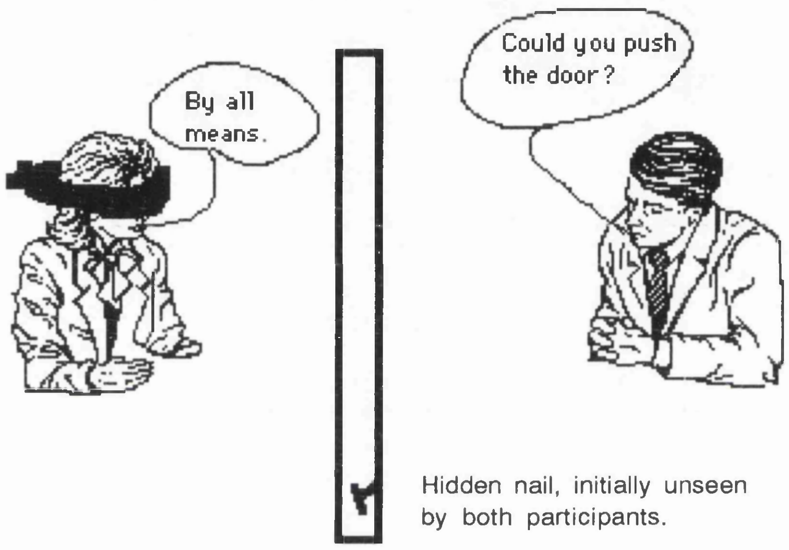


Figure 1.13 The task of getting through a closed door (conversation by humans, one blindfolded).

the way that the Artificial Intelligence joint planning theory currently proposes. This failure suggests that we should be looking for indirect coordination in terms of constraints. I shall be discussing this issue, together with a corresponding Artificial Intelligence solution, at the end of the thesis.

Chapter 2.

An empirical investigation. Goals in conversation: Increments of agreement and a measure of naturalness. [3]

[3] The theory of levels discussed in this chapter is explained in Draper, Oatley and Garrod (1987). A version of this chapter was submitted to Language and Cognitive Processes under the same title cited also here as Oatley, Draper and Button (in press). The main aim of including it here is to provide the reader with some insight into the problem of independently assessing naturalness in computer dialogue. It builds on the work of, for example, Hobbs and Evans (1980) since I indicate an explicit rather than subjective assessment of AI planning in relation to human dialogue transcripts. That is to say, it compares human dialogue with AI theory in a statistical fashion as opposed to a direct philosophical interpretation as they and others (e.g. Suchman 1987) did.

2.0 Introduction.

Demonstrating relationships between empirically observed human performance and psychological theory instantiated in artificial intelligence programs has always been problematic. Similarly, what actually occurs when people talk with one another has not always been closely reflected in the preoccupations of formal linguistics.

In this chapter I hope to contribute to a reduction of both these shortcomings by relating human conversation to a program of Power (1979) that simulates conversation. I do this firstly by offering some clarifying commentary on Power's work and its relation to other research on conversation and secondly by proposing that purposeful conversation generally can be understood in terms of a hierarchy of goals. Finally I describe an experiment in which people judged a transcript of a conversation using categories derived from my theory but which led to an outcome that I had not anticipated.

Three important components are commonly referred to in discussing the structure of conversation (a general account of other aspects to these components can be found in Clark (1985)).

The first is work on speech acts, in which it is claimed that utterances have an element of purpose. This research was initiated by Austin (1962) and Searle (1969) continued to develop this idea.

The second is Grice's (1957,1968,1975) proposal that conversation is coherent and based on a co-operative principle. Grice's maxims state that, when conversing, people must say about the right amount for the purpose in hand, be truthful, be relevant and be clear. These broad rules help to shape conversation. If they are not followed the conversation becomes tedious, or unacceptable.

The third is the idea of Sacks, Schegloff and Jefferson (1974), that in a conversation each person takes turns to speak. This has led to work on what a turn might consist of (e.g. Clark and Schaefer, 1987). Questions include how conversants yield their turn to the other person, how the end of a turn may be recognised, and so on.

Influential though all these proposals have been, each takes us only some of the way towards understanding how conversation is organised. So the proposal of Austin and Searle that utterances are purposeful is helpful, but it leaves us with an image of isolated speakers assuming that hearers have known properties and that they can be acted upon instrumentally. Grice's maxims indicate general rules about how to make an utterance useful, that is to say effective at achieving communicative goals, but they are stated in a very general way. Furthermore neither the theory of speech acts nor Grice's maxims tell us anything about how speakers co-operate, or what the structure is that holds a set of utterances together in a conversation. Sacks et al. identify the alternating contributions of participants but say nothing about what goals these contributions serve.

A partial solution to all these problems was offered by Power (1979) in an Artificial Intelligence program. He proposed that purposeful conversation is based on joint planning. In other words, when conversation is purposeful it is based on the construction of a plan, of which both speakers hold a copy. Speakers offer suggestions for construction or repair of any component of a developing plan. In this paradigm, conversation is a progressive sequence of moves, each establishing an increment of agreement in the process of constructing the plan, and with each person enacting those parts of the joint plan that can be performed during the conversation.

Power's decisive innovation was to conceptualise the atomic element of conversation not as a turn, not as a single utterance, and not as an act by which one person affects another, but as a set of utterances distributed between the conversants, each of which establishes an increment of agreement between them. Typically the heart of a unit is a pair of utterances, such as a proposal and an assent, a request and a reply, an announcement and an acknowledgement, and so on. The proposal thus combines a generalisation of the structural notion of two turns in an adjacency pair, the notion of utterances as purposeful actions aimed at achieving conversational goals, and a focus on co-operation, mutuality and agreement.

Power's proposal is, as it were, structurally deeper than the idea of taking turns. It is that if conversation achieves a purpose, i.e. an overall goal, then it can be understood in terms of the procedures by which sub-goals are achieved. Unlike planning by a single agent, however, sub-goals are achieved jointly. The structural notion of turn-taking thus becomes subordinated to the functional idea of describing how a purpose that can affect both conversants may be accomplished, when, in general, neither participant can accomplish this purpose alone. So one person may not know something. He or she may therefore ask the other, who may co-operate by providing an answer or by letting the first know that he or she cannot help. In such an exchange, what is accomplished is that the first speaker's goal of acquiring some knowledge is made explicit. At the same time the first speaker's goal is accepted by the other who may then help to fulfil it, or say that he or she cannot help. In completing the joint procedure the two conversants achieve a small increment of agreement between them.

The basic unit of conversation can, as Power suggested, best be called a 'conversational procedure'. The name draws on the computational idea of a procedure which is a coherent set of actions that, when invoked with the right preconditions, delivers a result which is the goal of the procedure. The qualifier 'conversational' indicates that the result is achieved jointly, and by means of speaking. A conversational procedure is co-operative. It requires two agents to accomplish it. One initiates and the other participates. The result achieved is an increment of agreement between the two participants in the conversation.

The program proceeds by having one or both of the agents pursue a specific goal within the context of initial conditions and beliefs. As an example, one agent might be given the goal of getting into a room, i.e. changing its state from Out to In, but be unable to change the state of the bolt because it is on the other side of the door. The agents then converse to negotiate goals, construct plans, exchange knowledge and act. A recognisable conversation arises naturally out of attempts by the agents to co-operate in a mutual plan. More fundamentally, basing the conversation around the building of a jointly acceptable plan is the most natural way of motivating adherence to Grice's principle of co-operation.

This idea has been extended by Houghton (1986). He considered that conversation, rather than being simply a set of moves by the speaker and hearer, was a series of interaction frames, comprising an initiation, an addressee and a monitoring of the uptake of what was said. In each frame there is knowledge about the interaction type which helps both the speaker and the hearer know what is expected within the conversational procedure. So, for Houghton, the unit of conversation is a conversational procedure within an interaction frame which, as in Power, can best be described by the type of conversational goal that it needs to serve. In Houghton, if one agent needs to know something or get the other to do something then it finds the corresponding frame to achieve it. What distinguishes Houghton from Power in this respect is that Power's agents do not reason about which conversational procedure to use or when it is appropriate (but see Cohen and Perrault 1979a,b), they just use it when needed and thus the meaning of a unit of conversation in terms of a conversational procedure is less clear. What is common to both definitions is that the effects of the exchange are analysed within the conversational procedure rather than as part of the main action cycle.

2.1 Levels of goals in conversation.

The theoretical proposal I wish to make is that Power's approach can be elaborated by structuring conversation in terms of a hierarchy of goals that the conversation achieves. My main suggestion is to distinguish between atomic conversational exchanges, and larger units that correspond to a topic. Power used conversational procedures of varying length for both of these.

My proposal is to make these levels of goals explicit, and to show how they are the basis for purposeful conversations generally. The levels of goals are as follows.

Level 1. Top level goals: purposes outside the conversation. When a conversation is purposeful, this means that participants have goals other than the conversation itself. The goals are the purposes they aim to achieve partly by means of the conversation. If any such external goals can be agreed, then these become top level goals of the conversation. In Power's case, such a goal might be getting one of the agents in through the closed and bolted door. A part of many conversations, therefore, will be the negotiation of which goals are to be agreed as mutual. The implication is that participants both join in the conversation to formulate plans that will achieve these goals. Thus in the simulation the agents might discuss whether they will jointly adopt the goal of getting agent John In.

Level 2. Topic sub-goals. The next lower level in a goal hierarchy concerns topics. Topic sub-goals emerge from the conversation through the initiation of one of the participants. A topic sub-goal is a component part of the overall plan tree, which can be initiated once a top-level goal has been agreed. It is defined in relation to the agreed external goal, and will be a means by which a step towards achieving it is attempted. If one agent were to suggest, for example, getting the door open, this would be an illustration of a level 2 sub-goal within the conversation.

Level 3. Sub-goals: minimal increments of agreement. At the next lower level is the atomic unit of conversation. It provides the functional basis for taking turns. This type of goal, then, is at the level in the goal hierarchy below the topic goals. I propose that a conversational procedure is the atomic unit because its result or goal is a minimal increment of agreement. Thus in a run of Power's program one agent may ask the other if it is In. The other replies: 'No'. Thus they accomplish a minimal increment of agreement about the state of the world. Sequences of such atomic conversational procedures are connected by means of the topic sub-goals. Each is initiated to construct parts of the plan tree dominated by a topic. Typically they occur in groups based on one topic at a time.

Level 4. Lowest level goals: sharing the airwaves. Whereas level 1 is about an external purpose, and levels 2 and 3 are about the contents of the conversation, this lowest level goal is about co-operating to use the medium of communication. Most obviously, two people should aim not to speak at once, or neither will be able to understand the other. This goal is a generalisation of the idea of taking turns.

2.2 The categorisation procedure.

If this theory of the structure of conversation is correct, then I should be able to take transcripts of purposeful conversations and analyse them to identify goals at these different levels. In this chapter, I concentrate on minimum increments of agreement. If this is a valid concept, then the first step would be to identify what constitutes such an increment.

I will also refer to conversational procedures as 'exchanges'. Broadly speaking they fall into the five groups shown below. The notation is as follows. The conversation involves just two participants, Speaker 1 who initiates the exchange, and Speaker 2 who participates in it. An utterance is something said by either speaker and it ends when the speaker stops or yields to the other. An exchange is a set of one or more utterances as indicated in square brackets [ ]. Here are the five groups, with examples.

(i) Utterance plus elision.

Speaker: [Utterance — No effect]

For example: ["Can you pick up some milk?" — ...]. For Speaker 1, the lack of reply is the most ambiguous of responses. It may or may not mean that the opening utterance has been received, and may or may not mean that it will become part of a plan.

(ii) Utterance plus a change in the state of the world.

Speaker: [Utterance — State change]

Here the utterance itself produces a change in the state of the world as in the kind of example offered by Austin e.g. ["I pronounce you man and wife"]. More frequent examples also occur. "I promise to bring the book back tomorrow" has the effect of changing the state of the world by creating a commitment.

(iii) Utterance plus an action.

Speaker: [Utterance — Action — State change]

Here an utterance is followed by an action which in turn changes the world, as in ["Pass the salt please"] — followed by the action, and its effect.

(iv) Two or more utterances involving discussion of some piece of information.

```
Speaker 1: [ Utterance —
Speaker 2:                  Utterance ]
```

This is the most easily recognisable, and perhaps commonest kind. An utterance of Speaker 1 is followed by an utterance of Speaker 2, for instance: ["It's raining". — "I'll be OK."].

(v) Two or more utterances involving the discussion of the result of some old piece of information (see Power 1984).

```
Speaker 1: [[ Utterance —
Speaker 2:                  Utterance ]
Speaker 1:                                  Utterance ]
```

In this kind of example embedding occurs, so that the opening utterance picks up a piece of information previously discussed, as in [["So we won't go to that movie after all." — "No"] — "All right then."].

Although exchanges often relate back to previous parts of a dialogue, the idea of the minimal increment of mutual agreement indicates that utterances are not necessarily linked semantically to one another. In the example given for group (iv), no amount of semantic machinery could link 'I'll be OK' to Speaker 1's utterance 'It's raining'. However, seen as an acceptance of a warning, or advice to take an umbrella out, Speaker 2's utterance delivers a perfectly acceptable result of a conversational procedure.

Like any procedure, conversational procedures are designed to exit with a result. Such results will be relevant to building or executing the mutual plan, which provides the linkages that hold the conversation together, and provides the basis for the coherence suggested by Grice's maxims.

As all increments are related to plan trees being built in the conversation, I can postulate that a small number of distinct categories of conversational procedure are possible, namely those that are relevant to planning. Conversational procedures achieve, as their result, an increment of agreement in one of the following components of planning. They relate either to Goals (G), Plans (P), Beliefs (B), or to what I call

Conversational Housekeeping (H) which is usually a remark serving to draw attention to, or comment on the intelligibility of the conversation itself, thus keeping it from going astray. I also find that some utterances in human conversation do not, apparently, contribute to a Level 1 goal in any way. These are not covered by the theory I am advancing. I call these miscellaneous (M). Figure 2.1 gives the definitions of each of

---

Category (G) Goal. In exchanges concerning a goal, participants talk about what they would like to achieve, although without necessarily saying by what plans they are going to achieve it. A goal is a state of mind, an effect that people would like to see happen in the world or in their own mind or in the other person's mind. Goals are achieved by plans and sometimes involve exchange of beliefs.
Category (P) Plan. Exchanges concerning a plan typically involve some question of what the people might do next, or how they might achieve some effect. Plans are in general about the future, and about how the world is to be adjusted to fit some goal. A plan may have just one step, or several. To make a plan, actors often discuss or negotiate ways of doing something before acting. Often it is necessary to think about the order in which actions must be performed to have a desired effect. Plans also sometimes involve considering actions which may not, in the end, be successful.
Category (B) Belief. Exchanges about beliefs concern information about the world in the present or past, either about general knowledge or things that can be known directly. They can be about what people think, suppose, remember or perceive about the world. Beliefs are essential to plans because plans can only be constructed on the basis of models of how the world works and the effects of actions. Questions of belief concern knowledge which may be true or false. Whereas a goal implies that the world should be changed by a plan to fit a state of mind, a belief is the opposite. It involves a person's state of mind that could be changed to fit the world in some way. Hence as well as beliefs being involved in discussions of plans, they may also be discussed as to their correctness.
Category (H) Conversational Housekeeping. Some conversational procedures are aimed at controlling the other person's participation in the conversation, for instance by indicating that a message has not been received. These exchanges are 'meta-content' in the sense that they have nothing directly to do with the topic under discussion.
Category (M) Miscellaneous. Hardly ever used but can be if the exchange does not fit into any other category.

<u>Figure 2.1. Definitions of five categories of conversational procedure (exchange).</u>

these categories. I have found that in natural conversation allowances must be made for exchanges involving more than one category, referred to as compounds (C). After considerable discussion and refinement of these concepts, and by applying them to transcriptions of conversations I have recorded, I decided to test whether they could be understood and applied by people outside my research group.

This test, which is like a reliability test in content analysis of text, will also serve to present the application of these concepts in actual dialogues. I will treat two transcripts, one a slightly augmented version of Power's task in which two human conversants discuss how to get one of them through a closed door and the other a discussion between two people on how to use a photocopier.

The following two conversations were transcribed to indicate exchanges, the speaker, the hearer, the action and changes in the state of the world. Wherever possible the coding also took into consideration elisions, interruptions, pauses, sounds, eye movements and gestures.

2.3 Getting in through a closed door task.

In Power's (1979) program two agents, in different rooms and separated by a door, talk to one another. On one side there is a bolt but the agent John has no means to open the door. From the other side the agent Mary can push the door but has no means to see. John's goal is to get into the other room with Mary's help. When humans are asked to act in such a scenario, the goal of getting in tends to be completed without speaking. In

Agent	Utterance	G/P/B/H/C	Action	Change
John and Mary decide to gain each other's attention in order to achieve John's goal of getting into Mary's room.				
1 JOHN:	[Mary.			
2 MARY:	Yeh.]	H		
John starts by executing a plan, sliding the bolt, and then asking Mary to assist him achieve his goal of getting in by pushing on the door. He sees it is not working, and asks why.				
			JOHN OPENS BOLT	BOLT OPEN
3 JOHN:	[Can you push on the door?			
4 MARY:	[Can I what?			
5 JOHN:	Can you push on the door, Mary?]	H	MARY PUSHES DOOR	NO CHANGE
6 MARY:	[Push the door?			
7 JOHN:	Yah...]	H		
	]	P	MARY PUSHES DOOR	NO CHANGE

Figure 2.1a Sample human sub-dialogue taken from the beginning of appendix 1 illustrating the closed door task.

order to test the theory of minimum increment of mutual agreement, I have extended this situation to include a large hidden nail running from the side of the wall into the door and to test it on human subjects instead of agents (see Figure 2.1a and appendix 1). I have also included a commentary on what the speakers' and hearers' various sub-goals and intentions were at each stage. In this dialogue there is no discussion about the goal which probably explains why Mary appears short-tempered throughout the dialogue. In theoretical terms, plans do not usually work unless there is a mutually-agreed goal. I used this transcript to train a group of subjects in categorising exchanges and then gave them the following transcript to test whether they were able to sort exchanges into those concerning goals, plans, beliefs and housekeeping.

2.4 Photocopying task.

In the human dialogue of figure 2.1b and appendix 2 John is seeking assistance from Mary to photocopy one piece of A4

Agent	Utterance	G/P/B/H	Action	Change
14 MARY:	[Pardon?			
15 JOHN:	It's alright. I'm just learning how to do it at the moment.			
16 MARY	Oh I see.]	G		
Now the rest is straightforward for both of them.				
17 JOHN:	[Right, so I put that down there. Right?			
18 MARY:	ahm...		CLOSES LID	LID CLOSED
19 JOHN:	And press that do I?			
20 MARY:	Yes.]	P	PRESSES START	MACHINE STARTS
	[Then it collects...			ONE SIDED COPY APPEARSBELOW
	and goes back you see.]	B		
	[Now now you turn it.]	P		
Having successfully completed a photocopy of the first side, John hastily prepares to press the start button again for the reverse side...				
21 JOHN:	[What I press again?		POINTS TO START	
22 MARY:	No, no. Well... ha ha ha.]	P		

Figure 2.1b Sample human sub-dialogue taken from the middle of appendix 2 illustrating the double-sided photocopying task.

paper with text on both sides. The same methods of categorisation and exchange marking are applied throughout.

2.5 Description of Experiment.

I now describe an experiment to see whether subjects can categorise exchanges in the photocopying transcript after some training with the transcript of getting through the closed door (see appendix 1 and figure 2.1a).

2.6 Subjects and procedure.

Eighteen final year undergraduates at Glasgow University were subjects. They were told that they would be asked to judge some human conversational exchanges, and to assign each to a category: Goal, Plan, Belief, Housekeeping or Miscellaneous. They were each given a sheet of paper containing the descriptions as shown in Figure 2.1. For each category, they were read out an example of an exchange that could plausibly have come from two humans negotiating the task of getting through the closed door.

Subjects were told that the experiment would be in two parts, the first being a training session of 45 minutes and the second being the actual experiment taking 15 minutes. The subjects were given a copy of the transcript of the task of getting through the closed door, as in appendix 1, with the exchanges marked but without the categories that I had assigned and without the annotations. Then an audio tape recording of the human conversation was played with subjects being asked to follow it in the transcript.

Next, subjects were asked to make up imaginary examples for each of the five categories of exchange for getting in through a closed door, to write them down in an order other than that given in Figure 2.1, and to record separately the category they had in mind for each example. Then, in pairs, subjects gave their examples to a partner, who was asked to categorise them. Subjects then compared their partners categories with their own, discussed each example, and came to a consensus where there were differences.

Next, I asked subjects to categorise a transcript of a conversation about getting in through a closed door, but not the one given in appendix 1. Again I asked subjects first to assign a category to each exchange, then to discuss it with their partners, and then to come to a joint decision. Next the experimenter ran through the transcript with the whole group of subjects, giving the categories assigned by the experimenters to each exchange, and discussing briefly all examples where subjects had assigned different categories than the experimenters.

Finally, in the training session, all the steps in the previous paragraph were completed using the transcript of appendix 1. I then removed all papers other than that with the categories (Figure 2.1). I distributed the transcript of the photocopying task (appendix 2), again with exchanges marked but uncategorised, and without annotations. I described the task briefly, played an audio-recording of the conversation and asked subjects, without consultation, to categorise the exchanges of the task.

2.7 Results of Experiment.

Overall, for the 19 exchanges in this transcript, the percentage agreement between subjects and ourselves was 51%, and respectively for each category: Goal 46%, Plan 50%, Belief 54%, and Compound (Plan-Belief) 52%. I did not assign any instances of Housekeeping in this transcript, and consequently there is no percentage agreement for this. The degree of agreement for each exchange in the transcript is shown in appendix 3.

A summary can also be made of the patterns observed when subjects coded categories that were not in agreement with my own choice of category. Figure 2.2 shows that there is some evidence that subjects do not disagree randomly but are coding certain categories. One explanation as to why subjects coded different categories was that they thought that the conversational exchange referred to a category that was higher up the goal hierarchy. This, to some extent, may explain why subjects chose different combinations of G,P,B,C:B,P from

Subjects

EXPERIMENTER

	G	P	B	C:B,P	H	O	# cases
G	46%	13%	0%	0%	20%	20%	54
P	0%	42%	22%	13%	12%	11%	144
B	0%	13%	51%	2%	14%	19%	90
C:B,P	2%	19%	32%	19%	5%	24%	54
H, O	-	-	-	-	-	-	0

Figure 2.2. Summary of appendix 3 showing the percentage agreement of subjects with the experimenter's categories and also the percentage frequency with which subjects recorded other categories. The last column shows the total number of cases (19 exchanges * 18 subjects = 342) considered for each category.

the experimenters. Differences of categorisation therefore came, not through subjects' misunderstanding of what they had been asked to do but because they genuinely felt that the exchange referred to another aspect of the main plan further up the goal hierarchy.

2.8 Discussion.

I interpret my results as modestly consistent with the hypotheses concerning the theory of conversation proposed by Power, and in terms of the hierarchy of goals that have been proposed. That I was able to obtain fifty one percent agreement, with a small amount of training and by people who had no

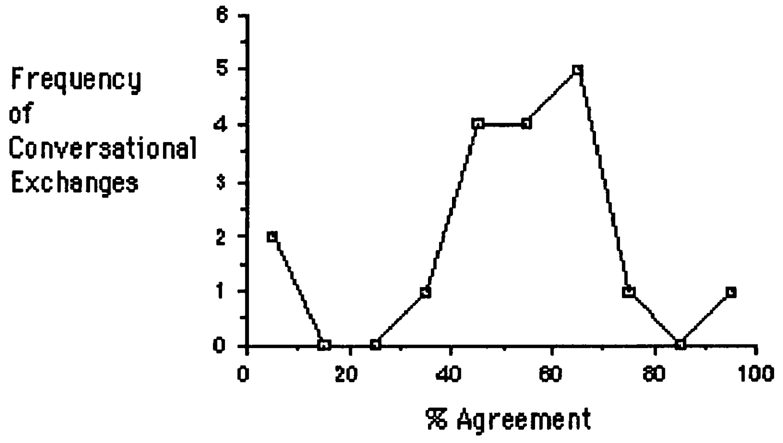


Figure 2.3 Graph showing the distribution of percentage agreement. The majority of the conversational exchanges (over 80%) could be categorised with over 40% agreement. Over half of the conversational exchanges could be categorised with between 50% and 70% agreement, the mean being just over 50%.

special expertise in making such codings, indicates that the categories I chose make some intuitive sense (see also figure 2.3). By comparison, highly trained raters (i.e. the experimenters who assisted with this experiment) of carefully constructed interview schedules would expect to achieve about 70% agreement on responses to items of an interview schedule.

There is one final point on the percentage agreement of

51%. Closer examination of the data shows that when subjects did not agree with the judgment of the research group they tended to go for similar categories which suggests that a cluster analysis may have been a more appropriate measure than simply a percentage agreement analysis. This would have measured not only the degree of agreement with the research group, as shown in figure 2.2, but also the level of agreement amongst subjects.

Some confusion obviously existed about the different categories. For example some subjects found it difficult to distinguish between a plan and a belief. One possible improvement might therefore be a further refinement in category definition. A better mutually exclusive set of categories might have been Goal, Causality, Propositional Content, Fact, Conversational Housekeeping. It would have also been useful to choose new transcripts in which the main physical goal was known by both participants in advance.

Purposeful human conversations can, to some extent, be thought of as being made up of exchanges, each of which achieves an increment of agreement about a goal, a plan or a belief, as discussed in the introduction to this chapter. I now move on to discussing some enhancements that I propose to make to Power's program that help to reinforce these findings.

<u>Chapter 3</u>.

<u>Connectedness in AI dialogue</u>.

3.0 Introduction.

In the previous chapter I showed that a joint planning theory, viewing dialogue as goal-directed, is a useful description of purposeful human conversation.

In this chapter, I discuss early work on computer dialogues that are not goal-directed and explain the progression to more recent developments. Particular attention is paid to describing computer systems that actually have a natural flow as opposed to those that demonstrate only one aspect of dialogue. In this chapter I include the non-goal oriented systems of ELIZA, Weizenbaum (1966), SHRDLU, Winograd (1972), PROTEUS, Davey (1978) that display monologues or one-sided dialogues together with techniques that could be used to go into monologues such as conceptual dependency, Schank and Reisbeck (1981). I then briefly look at alternative recent developments for goal-directed systems, e.g. Shadbolt and Musson (1987) and Carletta (1990) and techniques that go into the components of dialogue e.g. Cohen and Perrault (1979) in terms of conversational planning and Fikes and Nilsson (1971) in terms of physical planning.

I did not use any of the above, however, in the development of my own system SUPERPOWER since I was interested in considering a very simple domain with a very simple main goal and a much greater variety of knowledge and conversational skills. This explains why my main concern, however, is to move onto, and discuss, the complete dialogue system of Power (1979) which was the first goal-directed account of how joint planning provides a theory for the structuring of dialogue and is described in section 3.3. The systems of Houghton (1986,1989), Houghton and Isard (1987) and Power (1974,1979) are then discussed in a detailed comparison in section 3.4. I then finish the chapter by explaining how SUPERPOWER (section 3.5 and chapters 4 and 5 listed in the appendix) is an enhancement on these latter two systems.

3.1 Early AI models of dialogue.

In this section I focus on systems that are more about monologue, one-sided dialogues, sub-dialogues and the problem of how to interpret appropriate techniques that can be applied to these special types of purposeful conversation. Often these systems can not interact with a copy of themselves and there is no concept of a mutual plan which is often left up to the human user.

ELIZA[4], Weizenbaum (1966), was the first system that demonstrates connectedness in conversation. It is an attempt to imitate a psychotherapist interacting with a patient in a therapeutic conversation similar to those described by Freud (1904), Jung (1910) and Rogers (1951). It has absolutely no knowledge of the world and thus, by concealing its lack of understanding, Weizenbaum can simulate the activity of a counselling session (see figure 1.5). In this system he shows how repetitions can be avoided by randomly choosing alternatives. He demonstrates a kind of naturalness (chapter 2 in this thesis), since some subjects, when confronted with the program, have found it hard to believe it originates from a machine.

In contrast to ELIZA, Winograd's (1972) SHRDLU [5] knows a great deal about its domain and which consists of a room full of blocks of different shapes and sizes. The user can issue a multitude of instructions and commands such as "Put the big blue block on the big red block", or questions such as "Is the green block on the red block?". However, although the discussions are always correct, the dialogue is severely limited

[4] PARRY (Parkison, Colby and Faught 1977), a paranoid agent that could engage in conversation with a psychiatrist was an extension of the same principles behind ELIZA except that a more generalised pattern matcher was used to pick out a specific connotation of the utterance rather than its syntactic form. Connectedness was achieved by interpreting every utterance in a paranoid way.

[5] Understanding discourse and text has also been treated as the same thing in Psycholinguistics; see for example Sanford and Garrod (1981).

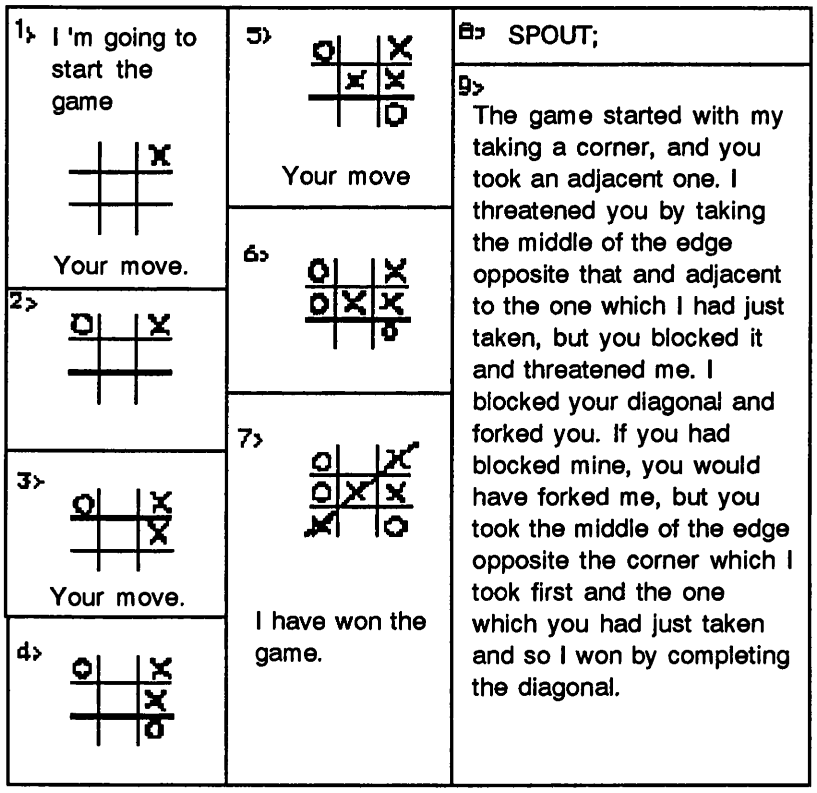
by the syntax, semantics and planning algorithms of the small domain. Connectedness is demonstrated by the program's ability to remember where all the blocks are, to respond with an appropriate action or utterance, and finally to use the correct pronoun reference[6]. Figure 1.6 illustrates how Winograd's stimulus response mechanisms appear to have more credibility than those offered by Weizenbaum in that the program exhibits sensible behaviour within the limited world that it knows about.

SHRDLU is limited in its simulation of dialogue structure (i.e. there are exchanges but none of the exchanges relate to one another and are randomly chosen by the human user). This is due to the absence of an organised structure of goals for the dialogue. Winograd's program does not address the issue that dialogue is not simply about ordering an agent to do things by choosing random goals but that it also involves co-operative activity. Nor is dialogue about choosing random responses (as Weizenbaum) but on the contrary it has to have an element of structure in its execution. Winograd does however achieve connectedness in terms of what the system knows and can do in the world, as illustrated in Figure 1.8. The element of structure in dialogue (e.g. a continuing plan) is the main idea in this

[6] Anaphoric reference is not discussed in detail here, but see for example Hirst (1981). Garnham (1985) discusses some of the deficiencies of Winograd's program in this respect.

thesis which I shall be taking up in more detail in later chapters.

Another strand of work has considered the problem of how it is possible for a system to explain itself.[7] PROTEUS, Davey (1978), was one of the first attempts at describing a



1> I 'm going to start the game

Your move.

2>

3>

Your move.

4>

5>

Your move

6>

7>

I have won the game.

8> SPOUT;

9>
The game started with my taking a corner, and you took an adjacent one. I threatened you by taking the middle of the edge opposite that and adjacent to the one which I had just taken, but you blocked it and threatened me. I blocked your diagonal and forked you. If you had blocked mine, you would have forked me, but you took the middle of the edge opposite the corner which I took first and the one which you had just taken and so I won by completing the diagonal.

Figure 3.1 PROTEUS, Davey (1978). Connectedness in terms of explanation.

system that generates sentences in a principled manner. This problem was addressed by Winograd, but only partially (see

[7] As an alternative to AI approaches to speech production, Harley (1984) collected samples of errors occurring in everyday speech and by analysing word frequency he was able to develop his own top-down theory of speech production. For an overview of other methods such as pauses or hesitations within Psycholinguistics see Garnham (1985: pps 183-204). Butterworth (1980) also presents a variety of attempts at modelling speech production including a discussion about early AI work, see also Steedman and Johnson-Laird (1980: pps 111-139).

Davey 1978: pp 33), and also by Weizenbaum, but only in the form of pattern matching without real understanding. After providing an exhaustive account of very early AI systems Davey goes on to show how PROTEUS (figure 3.1) can generate dialogue to explain the rationale behind its moves in a game of noughts and crosses. It does so by retaining a plan representation of how it arrived at each move during the game. Thus, having decided on an appropriate plan or when the game was complete, the system can then use a systemic grammar, Halliday (1961) to decide on an appropriate sentence to construct (Davey, 1978, Houghton, 1986). The system could therefore explain its choice of moves at any point during the course of the game. In the example chosen, the program's rational explanation for its choice of moves is given when the game has finished, thus illustrating some aspects of a monologue. It could also have discussed things during the game, thus allowing the user a chance to learn and make a better response, thus illustrating some aspects of a dialogue.

The diagram illustrates that there is a flow in the dialogue which has either an element of conversation, demonstrated by its ability to recall all aspects of the changing world, or an element of stimulus response, i.e. there is an action associated with the interaction, as described in previous systems. This flow is similar to the SHRDLU program of the preceding section except that, whereas Winograd focuses on the problem of understanding questions, Davey concentrates on how the hearer might generate intelligent replies.

There are some systems reporting a kind of dialogue that is not goal-oriented and much more like a monologue. That is to say, the dialogue is often one-sided. The justification for this is that early AI dialogue systems were not based on sound

1. COMPUTER:	Do you like chips ?
2. LEARNER:	Yes.
3. COMPUTER:	Do you like kiwi fruit ?
4. LEARNER:	What is a kiwi fruit?
5. COMPUTER:	It has a brown skin.
6. LEARNER:	Yes.
7. COMPUTER:	Do you like mango ?
8. LEARNER:	Yes.
9. COMPUTER:	Do you like pomegranate ?
10. LEARNER:	Yes.
11. COMPUTER:	Do you like papaya ?
12. LEARNER:	No.
13. COMPUTER:	Can you make biscuits?
14. LEARNER:	You roll out some pastry. You get some shapes. You cut out the shapes of pastry. You put them on a square thing. You put them in the oven.
15. COMPUTER:	Where did you get the pastry from?

Figure 3.2 Button (1989). Connectedness in terms of subject matter.

semantics and in order to understand natural language a computer has to not only syntactically understand a sentence but also interpret it in the light of what it already knows.[8] One

[8]These ideas stemmed from Bartlett (1932) who described memory as an active organisation of past reactions. For a discussion, in Psycholinguistics, about the consensus of opinion as to how many different memory systems there are, see Baddley (1976) and for the critique see Crowder (1982).

idea along these lines was developed by Schank (1972), Schank and Reisbeck (1981) and Schank (1980,1982,1985). According to Schank, in order to understand a concept it must be related to another more basic concept. He developed a theory of conceptual dependency which involves a number of primitives into which all sentences can be broken down.[9] This leads to a number of attempts at story understanding that meet with limited success but account for monologue or one-sided dialogue. In my own system, described in Button (1989), Button, Oatley and Draper (1989) and illustrated in figure 3.2, I show how a simple discussion about eating habits can be modelled using Schank's Conceptual Dependency theory. Connectedness is achieved here through the regularity of the subject matter, thus allowing direct access to simple memory structures such as those described by Schank.

Useful techniques for monologues were also explored in other areas. For example, intelligent continuity can occur if, within a given domain, an appropriate knowledge representation such as Conceptual Dependency (Schank 1980), Scripts (Schank and Abelson 1977), Frames (Brown 1987), Semantic Networks (Collins and Quillian 1969) or Logic (Cloksin and Mellish 1981, Turner 1984), is used. Indeed some of these AI techniques have

[9] These sorts of ideas have often been compared with the principle of compositionality proposed by the 19th century philosopher Frege.

been applied successfully to one-sided dialogue. For example, Grosz and Sidner (1986) show how a semantic network can be used to define the focus of dialogue either in terms of the task complexity or in terms of the conversation[10]. GUS (Bobrow, Kaplan, Kay, Norman, Thompson and Winograd, 1976) shows how one-sided dialogue can be driven by the use of a frame. All these techniques, without doubt, offer models of how humans might conduct conversation if the nature of the main goal of the conversation is fixed in advance and the task does not consist of a mutual discovery of how to realise the goal.

There is work which has considered parts of a one-sided sub-dialogue in the form of question-answering systems (for a full explanation see Webber 1987). Allen and Perrault (1980) and Allen (1983,1987) showed that by analysing the intentions expressed within an utterance, nested conversation (see examples from previous chapter in which conversational exchanges can be thought of as embedded) can take place. They then follow this up by illustrating it with two types of conversation: (a) that which moves towards clarification, as in figure 3.3:

[10] Within the field of Psycholinguistics this is also discussed in terms of how conversation is focused. For a full explanation see, for example, Garnham (1985), Sanford and Garrod (1981) and Grosz (1977) .

1. HUMAN:	When is the Windsor train?
2. COMPUTER:	To Windsor ?
3. HUMAN:	Yes.
4. COMPUTER:	3.15.

Figure 3.3 Allen and Perrault (1980) Connectedness in terms of dialogue clarification.

and (b) those that are driven by dialogue failure as in figure 3.4:

1. HUMAN:	When is the Windsor train?
2. COMPUTER:	It leaves at 3.15
3. HUMAN:	No, I want to meet the Windsor train.
4. COMPUTER:	Ah, in that case it arrives at 3.10.

Figure 3.4 Allen and Perrault (1980) Connectedness driven by dialogue failure.

Allen et al. improve the robustness of the system through the disambiguation of vaguely specified plans. Thus, they are able to show a more detailed account of a theory of indirect speech acts than that offered by Searle's (1969,1975) theory that I briefly mentioned in Chapter 2.

This now completes my analysis of the body of work that has described dialogue using non goal-directed techniques. As Power (1979) correctly observes, the main problem with early AI work is that none of the programs developed can converse with a near copy of itself and thus can hardly be considered a model of purposeful dialogue. From now on I shall only be discussing goal-directed systems.

3.2 Recent developments.

In this section, I look at studies that have examined the components of a true dialogue system by which I mean a dialogue system that recognises the need for conversation to be based in some way on a mutual plan.

As will be seen the problem of this focus on the components of dialogue is that the systems created are not actually able to generate English sentences. Some work has focused on either speaking or listening but not both. For example Appelt (1985), Cohen and Perrault (1979a,b), Pollack (1986), Cohen and Levesque (1985), and others show all the components of what goes into an utterance during the course of a dialogue about a mechanical task. In particular, Cohen and Perrault have produced a plan-based definition of a speech act by considering what it is the speaker really wants and what he is trying to say. Litman and Allen (1987) and Allen and Perrault (1980) show methods for understanding underlying plans occurring in an utterance.

Some studies have considered how to plan an overall task within a single agent domain by looking at how to plan actions that achieve a particular goal. Overviews of all the many different planning systems can be found in Tate (1985) as a technical description, in Steel (1987) as a beginner's guide and in Agre (1990) as a philosophical argument for and against comparing these methods with those of human reasoning.

I shall be referring to this approach as **classical planning**. Some work is particularly prominent in this area and should be mentioned. STRIPS, Fikes and Nilsson (1971) was the first system to formalise and plan around actions, preconditions and effects. MOLGEN, Stefik (1980,1981) looks at the need for meta-planning. ABSTRIPS, Sacerdoti (1974) Freidland and Iwasaki (1985) looks at abstract plan representations before developing a total plan. NONLIN, Tate (1976) uses a goal structure. SIPE, Wilkins (1984) looks at domain-independent planning. The essential feature of all these systems is that they are trying to plan for a particular goal by chaining through plan operators or causal rules in a specific way.

More recent work has studied the problem of reacting to unexpected changes in the state of the world, what Suchman (1987) refers to as plans and situated action and what I shall be referring to as **unexpected planning** in future chapters. These works include Lansky (1988) on the distinction between events and processes, Ginsberg and Smith (1986) in terms of reasoning about action, and Wood (1990) in terms of planning in a rapidly changing environment.

Planning areas not covered by any single agent planners can be divided into two types. Firstly there is what I shall be referring to as **instrumental testing** in which agents are required to perform tests in order to infer new facts about the world. Secondly, there is **experimental planning** in which agents are required to try actions and see what happens in order to learn about the consequences of a change in the states of the world. These categories of planning have sometimes been referred to as replanning and have taken their name in order to deal with those circumstances that arise when the main planning routine breaks down. I shall be discussing these four fundamental kinds of planning (classical, instrumental, unexpected and experimental) at the end of the chapter and again in chapter 5 and show how they can be integrated into the main action cycle.

In short, using single agent planning techniques causes serious difficulties in 2-agent dialogue planning since a good planner needs to be not only good at the job but also flexible and explicit in conversation.

More recently, research has focused on discussing the problem of how to communicate the underlying intentions of the agents. Shadbolt (1984), Shadbolt and Musson (1987) and Carletta (1990) showed how efficiency (see section 1.1 for precise definition) can be achieved in conversation by using what Shadbolt refers to as a set of communicative postures. Utterances are then chosen through the use of Game Theory (e.g. Charniak and McDermott 1985). The underlying plans are then passed from one agent to the other. Postures can also have different settings so as to allow different levels of communicative competence, hence permitting an interpretation of explicitness and accounting for different dialogue styles. One of the difficulties with having a large number of alternative settings for a simulated agent is that it is difficult to know whether or not all combinations of these parameters in fact give rise to successful dialogue. Furthermore it then becomes an issue as to how to interpret them. Systematic parametric studies are needed to test this, although none have been reported so far. One way of solving this problem is to illustrate the dialogue with a comprehensive "Monte Carlo" simulation rather than an exhaustive one. That is to say, a selection of important initial dialogue settings are chosen as opposed to all settings (see chapter 4).

For instance, in figure 3.5, Shadbolt and Musson (1987) illustrate how connectedness can also be achieved by combining more than one goal in an utterance. However, the more efficient the computer dialogue the more difficult it becomes to evaluate its robustness under variation.

1.	INSTALLER:	I'm going to lay the storm drains.
2.	DECORATOR:	Could you install the rough plumbing instead?
3.	INSTALLER:	No. I will have to install the drains first.
4.	DECORATOR:	OK
5.	INSTALLER:	I have installed the drains.
6.	DECORATOR:	OK
7.	INSTALLER:	I have installed the rough plumbing.
8.	DECORATOR:	OK. Could you install the rough wiring?
9.	INSTALLER:	OK. I have installed the rough wiring.
10.	DECORATOR:	OK. Could you install the air conditioning?
11.	INSTALLER:	No. You will have to pour the basement floor first.
12.	DECORATOR:	OK. I have poured the basement floor.
13.	INSTALLER:	OK. I have installed the air conditioning.

Figure 3.5 Shadbolt and Musson (1987). Connectedness in terms of communicative postures, multiple goals and efficiency by constraint satisfaction. (Simplified from dialogue about plan structures).

In short, an efficient dialogue about a complex joint task that generates utterances based on constraints has an element of naturalness but is too mechanistic and regular to be believable. Nevertheless, it is an appropriate model when both participants know exactly what they are doing and their knowledge is similar. What efficient models of conversation cannot demonstrate is why irregular frills occur in human dialogue. There are four main reasons: (1) humans think at different speeds from one another, (2) they conduct conversation in different ways from each other, (3) they instrumentally manage topics in different ways, (4) they engage in conversation with different goals. It is for this reason that I turn the reader's attention to systems that separate physical and conversational planning.

I hope to show these questions can be answered by keeping both the domain and main goal simple and varying the knowledge and conversational skills of the agents. I now discuss the work of Power and Houghton as these are the only other two systems to have done this.

3.3 Power's robot world.

In contrast to the work discussed in section 3.1, Power (1979) uses a simulation of computer-to-computer interaction and is able to demonstrate how conversation is structured as a whole.

Power's simulated dialogue concerns interactions between two agents who are on either side of a door locked by a bolt. In this simple situation, illustrated in figure 3.6 and in earlier

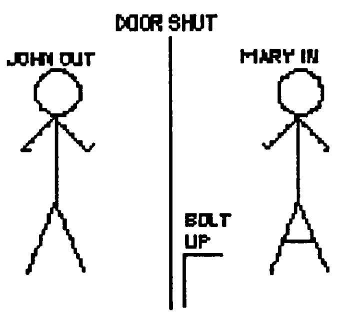


Figure 3.6 Power's robot world.

chapters, there are only four objects, each of which can be in one of two positions. There are the two agents, called John and Mary. Each agent can be either In or Out. The door can be Open or Closed. The bolt can be Up or Down. To explore the interactions of the agents, the program can be started with different initial conditions: that is, each agent can know or be ignorant of certain laws of nature, or have a false belief about them. An agent can acquire or change a belief if informed or persuaded by the other, or if, after an action, something unexpected happens. Agents can differ in their perceptual abilities — for instance one may be able to perceive the state of the world, and the other may be blind. In this way various aspects of cognition such as certain kinds of perceptual ability, or knowledge, can be distributed between the agents.

This, in turn, helps to simulate those ubiquitous human situations in which an agent cannot accomplish certain actions or achieve certain results in isolation, but must co-operate with another person to do so.

Connectedness is achieved through what Power refers to as conversational procedures (CP)[11] which allow the agents to co-operate with one another to achieve predefined goals. Furthermore the course of the dialogue can be directed by the use of simple physical causal rules (for example the bolt must

[11] For the sake of brevity, and for future reference, I shall refer to a conversational procedure as a CP. In the case of Houghton I shall refer to an interactional frame as an IF.

be Up to open the door) that the agents use to construct a joint plan tree. Planning and execution is then represented in the form of a control stack which in some sense represents the point of the utterance at any time during the dialogue. The dialogue, however, lacks efficiency since the conversational procedures themselves appear as utterances.

These problems of efficiency were taken up by Houghton (1986) and will be discussed in more detail later. He divides the problem into three levels i.e. Utterance planning (Davey 1978), Conversational planning (Cohen and Perrault 1979a,b) and Physical planning (Fikes and Nilsson 1971) and is thus able to generate speech for this domain in a more natural way. Houghton's is the second principal significant body of work which will be closely studied, along with that of Power, in order to bring me to SUPERPOWER. These two systems are the concern of the next section.

3.4 A comparison between Power and Houghton.

Every dialogue system must be structured according to a number of important components. What these components are is still an open-ended question since a lot depends on the domain of the dialogue system being modelled. In this domain, where the agents are required to negotiate co-operatively through one agreed goal, the components fall into a number of categories. Agents must produce dialogue and also produce and understand

	Power	Houghton	SUPERPOWER
Dialogue Content. (3.4.1)	Over Explicit.	Efficiency, Indirect and direct speech acts.	Explicit. Dialogue stretches over several actions.
Language Generation. (3.4.2)	Unprincipled.	System Network and GPSG	Unprincipled.
Language Comprehension. (3.4.3)	Unprincipled.	Does not exist.	Unprincipled.
Physical Planning. (3.4.4 & ch's 5,6)	Fikes and Nilsson (1971) i.e. Uses causal rules.	Fikes and Nilsson(1971).	Uses causal rules, exhaustively but hardwired.
Conversational Planning. (3.4.5 & ch's 5,6)	7 detached procedures.	4 Interactional frames Cohen and Perrault (1979a,b).	6 detached procedures.
Meta Planning. (3.4.6)	Distinguishes between an event and a situation.	Similar to Power.	Similar to Power but the time slicing considerably assists this.
Executive Control. (3.4.7 & Ch 4)	Control passed alternately after each functional unit.	Uses pop11 processes.	Time slicing and the notion of expecting a reply. Definition of an increment of time.
Initial settings and representation. (3.4.8)	Some straightforward lists of data.	Some structure e.g. know (fred is in)	Knowledge can be known, given, told, unknown, estimated, inferred or undefined.
Conflict resolution. (3.4.9)	Hardwired in CP.	Similar to Power	Similar to Power, but with extra facilities to deal with interruptions.
Interaction skills. (3.4.10 & Ch's 4 ,5)	CP's , priming. e.g. "May I ask you something?" Can handle partially embedded conversation, but the same conversational procedure cannot be active twice. No embedding.	IF's, no priming. Can handle embedded conversation but not more than one action.	Similar to Power plus extra planning facilities. CP's, priming, time slicing Topic has as subgoal, an action and the successful completion of the subgoal. No reasoning abilities, multiple CP's and actions are dealt with.
Inference (3.4.11 & Ch 5)	Hardwired in conversational procedure.	Hardwired in physical planner not exhaustive.	Hardwired in physical planner but exhaustive.

Figure 3.7 Comparison of the main differences between Power (1979), Houghton (1986) and my own system SUPERPOWER.

utterances. In order to do this they need a planning algorithm to support conversational and physical action. How this is structured is crucial to all dialogue. Finally, there must be some kind of meta-planning and coordination to deal with the cases where agents are required to make choices between competing courses of action. Knowledge representation is needed at source for the sake of program clarity. Topic structure is also needed to progress instrumentally from one sub-goal to another since to avoid doing so would lead to pragmatic ambiguity. Figure 3.7 summarises the main differences between Houghton, Power and my own system SUPERPOWER. Thus the next series of sections discuss the differences under these headings that reflect the major components that must go into any dialogue system.

3.4.1 The dialogue content.

It is by no means clear how a dialogue system should present itself. On the one hand a theory provides evidence of dialogue principles but it does not systematically demonstrate the claim in terms of natural utterances (e.g. Searle 1990, Cohen and Perrault 1979a,b, Cohen and Levesque 1990), while on the other hand dialogue generation without principles does not lead us to dialogue understanding (e.g. ELIZA Weizenbaum1966).

However, a great deal can be observed from the computer dialogue provided it is represented in a way that reflects the theory. One way of overcoming this is to provide a linguistic interface that displays the theoretical features. For example

(STATE OF WORLD IS NOW [JOHN OUT, MARY IN, BOLT UP, DOOR SHUT])

1 JOHN: MARY.
2 MARY: YES.
3 JOHN: I WANT TO SUGGEST A GOAL.
4 MARY: GO AHEAD.
5 JOHN: WILL YOU HELP ME GET IN ?
6 MARY: BY ALL MEANS.
7 JOHN: SHALL WE MAKE A PLAN ?
8 MARY: JOHN.
9 JOHN: YES.
10 MARY: MAY I ASK YOU SOMETHING ?
11 JOHN: GO AHEAD.
12 MARY: ARE YOU IN ?
13 JOHN: NO.
14 MARY: SHALL WE MAKE A PLAN ?
15 JOHN: OK.
16 MARY: JOHN.
17 JOHN: YES.
18 MARY: MAY I ASK YOU SOMETHING ?
19 JOHN: GO AHEAD.
20 MARY: CAN YOU MOVE ?
21 JOHN: YES.
22 MARY: JOHN.
23 JOHN: YES.
24 MARY: MAY I ASK YOU SOMETHING ?
25 JOHN: GO AHEAD.
26 MARY: IS THE DOOR OPEN ?
27 JOHN: NO.
28 MARY: I SUGGEST THAT WE GET THE DOOR OPEN AND THEN YOU MOVE..
29 JOHN: MARY.
30 MARY: YES.
31 JOHN: I WANT TO EXPLAIN SOMETHING.
32 MARY: GO AHEAD.
33 JOHN: IF I MOVE, NOTHING HAPPENS.
34 MARY: I DISAGREE. IF YOU MOVE WHEN THE DOOR IS OPEN, YOU CHANGE POSITION.
35 JOHN: I SEE.
36 MARY: I SUGGEST THAT WE GET THE DOOR OPEN AND THEN YOU MOVE.
37 JOHN: ALL RIGHT.
38 MARY: SHALL WE MAKE A PLAN ?
39 JOHN: OK.
40 MARY: I SUGGEST THAT I PUSH THE DOOR.
41 JOHN: ALL RIGHT.

(STATE OF WORLD IS NOW [DOOR OPEN, JOHN OUT, MARY IN, BOLT UP])

42 MARY: I WANT TO TELL YOU SOMETHING.
43 JOHN: GO AHEAD.
44 MARY: I HAVE PUSHED THE DOOR.
45 JOHN: I SEE.
46 MARY: LET'S ASSESS THE RESULT OF MY ACTION.
47 JOHN: OK.
48 MARY: NOTHING HAS HAPPENED.
49 JOHN: MARY.
50 MARY: YES.
51 JOHN: I WANT TO TELL YOU SOMETHING.
52 MARY: GO AHEAD.
53 JOHN: THE DOOR IS OPEN.
54 MARY: I SEE. THE DOOR HAS CHANGED POSITION.
55 JOHN: YES.
56 MARY: THE DOOR IS NOW OPEN.
57 JOHN: RIGHT.

(STATE OF WORLD IS NOW [JOHN IN, MARY IN, BOLT UP, DOOR OPEN])

57 JOHN: I WANT TO TELL YOU SOMETHING.
58 MARY: GO AHEAD.
59 JOHN: I HAVE MOVED.
60 MARY: I SEE.
61 JOHN: LET'S ASSESS THE RESULT OF MY ACTION.
62 MARY: OK.
63 JOHN: I HAVE CHANGED POSITION.
64 MARY: JOHN.
65 JOHN: YES.
66 MARY: I WANT TO TELL YOU SOMETHING.
67 JOHN: GO AHEAD.
68 MARY: YOU ARE OUT.
69 JOHN: I DISAGREE. I HAVE CHANGED POSITION.
70 MARY: YES.
71 JOHN: I AM NOW IN.
72 MARY: RIGHT.

Figure 3.8 Power (1979) Connectedness in terms of joint planning and goal-oriented processes.

the distinction between "The bolt must be Up" and "the bolt is Up" reflects on the surface, and in the dialogue, evidence that the program is using a theory of inference by way of a linguistic method such as modality (Palmer 1990, Coates 1990, Stubbs 1986). In Power there are some utterances about the type of conversation that is about to follow and so the dialogue demonstrates his theory about conversational procedures but without concern for efficiency. In Houghton features such as direct and indirect speech acts are demonstrated as an efficient interaction.

An example of the dialogue generated by Power's system can be seen in figure 3.8. Notice that every main exchange is primed[12] with another exchange such as "I want to explain something" so as to allow agents to be absolutely sure what the subsequent exchange refers to. This has the effect of making the dialogue appear over explicit.

In Houghton's system the dialogue for a similar goal is more efficient and appears to correspond to how humans would conduct such a conversation. There are a number of distinguishing features to note about the two dialogues shown in figures 3.8 and 3.9: (i) Houghton's program can generate all of Power's dialogue but without using the priming of utterances such as "Shall we make a plan?" or "I want to suggest a goal".

[12] Using two pairs of utterances to convey one message. As in "I want to explain something" "By all means" "If you move when the door is shut nothing happens" "Right". Priming, however has a much wider definition within pragmatics and is normally used throughout dialogue to assist the hearer in knowing what to expect next.

(ii) It distinguishes between indirect and direct speech acts (as in lines 11 and 13). (iii) It knows what to do when one and only one of the two agents can not think of a plan (as in lines 6-9).

1. Fred:	Doris.
2. Doris:	Yes.
3. Fred:	I want to be in.
4. Doris:	I see.
5. Fred:	Could you push the yellow door.
6. Doris:	No, because the bolt isn't up.
7. Fred:	Is there a bolt that is up ?
8. Doris:	No.
9. Fred:	How do you get a bolt to move ?
10. Doris:	You get to be in, then you slide it.
11. Fred:	Could you slide the green bolt.
12. Doris:	OK.
13. Fred:	Push the door.
14. Doris:	OK.

Figure 3.9 Houghton (1986). Connectedness in terms of efficiency and natural language generation.

However neither of these systems are able to generate sub-dialogues that relate to the consequences of an action through the use of inference. Power has one conversational procedure (game **ZGASSESS)** but no clear inference within the planning process. Furthermore neither system can deal with the cases when both agents are stuck for (a) a plan, (b) a piece of knowledge or (c) an unexpected event. Thus the choice of who speaks next is largely determined by the physical goal and knowledge that the agents are given at the outset rather than by any differing conversational skill.

3.4.2 Language Generation.

In Power the language generation mechanism is *ad hoc.*

The system generates sentences by matching the plan description to the utterance in a one-to-one fashion. Such sentences contain a subject, a verb and an object, or alternatively, in a discussion about a belief, the utterance will have a parallel structure of subject, verb and object or an "..if..then..." clause. Two sets of exchanges are uttered, one relating to an appropriate exchange type and the other to its content. Examples include:

1) Ist exchange contains **zggoal**[13] and utters "I want to suggest a goal." 2nd exchange contains the goal [John In] and utters "Will you help me get In?"

2) Ist exchange contains **zgask** and utters "May I ask you something?" 2nd exchange contains the question [Is John In] and utters "Are you In?"

3) Ist exchange contains **zgplan** and utters "Shall we make a plan?" the second exchange contains the plan [undef] and asks a few questions until a plan has been found.

4) Ist exchange contains **zgrule** and utters "I want to explain something." 2nd exchange contains the causal rule [evt [robot push] sit [any] res [nothing]] and utters "If you push the door nothing happens."

[13] zggoal, zgplan, zgask, zgrule, zgtell, zgassess and zggame are the pop11 function names of the seven conversational procedures described in Power (1974,1979). SUPERPOWER, described in chapter 4 and 5 uses six of these and zgassess has been removed and combined with zgtell and a new inference that assesses the consequences of an event.

5) Ist exchange contains **zgtell** and utters "I want to tell you something". 2nd exchange contains the action [Mary push] and utters "I have pushed the door".

6) Ist exchange contains **zgassess** and utters "Let's assess the consequences of my action." 2nd exchange contains the assessment of the action [undef] and utters "Nothing has happened."

In all cases utterances are mapped from a plan structure in an unprincipled one-to-one fashion.

In Houghton's program the production mechanism is more principled and uses system networks and a Generalised Phrase Structured Grammar, see Houghton and Isard (1987:12-15), a network with a lexicon, and morphological and syntactic representations to convert a semantic description into an English expression. He avoids priming in exchanges and generates what is to be said in one exchange. The following information passes to the language production mechanism: the initiator, addressee, name of the interaction frame and semantic content. Examples similar to the above might be:

1) Doris, Fred, GET_TO_DO[14] , [push Doris door2]: would generate the sentence "Could you push the yellow door?"

2) Doris, Fred, FINDOUT, [atloc Fred in]: would generate the

[14] GET_TO_DO, FINDOUT, MAKEKNOWN and GET_ATTENTION are the pop11 function names of the four interactional frames described in Houghton (1986).

sentence “Are you In?”

3) Doris, Fred, MAKEKNOWN, [havegoal Doris [atloc Doris in]]: would generate the sentence: “I want to be In.”

4) Doris, Fred, MAKEKNOWN, IF <action movethru robot door> WHEN [door open] THEN [move robot]: would generate the sentence “If you go through a door when it is Open then you move.”

5) Doris, Fred, MAKEKNOWN, [BECAUSE [atloc bolt2 down]] would generate the sentence “No, because the bolt isn't Up.”

This differs from Power in that grammatically correct sentences can be generated in a principled manner from a structured plan. It also takes into consideration current topics.

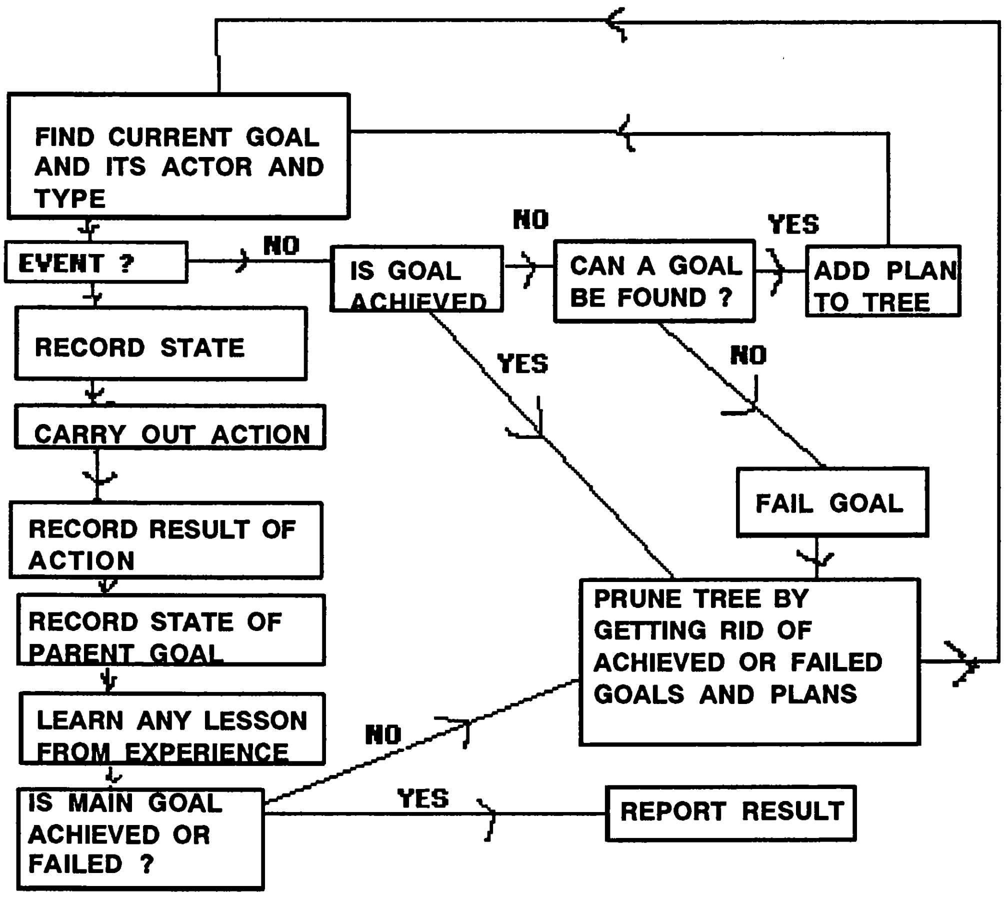
3.4.3 Language comprehension.

In Power’s program there is a simple parser that maps in an unprincipled way from an utterance onto a plan structure that is then passed with relevant factual information onto a conversational game. As there is only a limited number of sentences (approximately fifty), choices about which plan content matches which sentence are relatively easy.

There is no mechanism for this process in Houghton's program. Instead the plan content is passed directly from one agent to the other without parsing the sentence itself.

3.4.4 Physical Planning.

Power splits the planner into two components; one for achieving a goal and the other for finding a plan. In order to achieve a goal the system may look for a plan, carry out an



FIND CURRENT GOAL AND ITS ACTOR AND TYPE
EVENT ?
NO
IS GOAL ACHIEVED
NO
CAN A GOAL BE FOUND ?
YES
ADD PLAN TO TREE
YES
NO
RECORD STATE
CARRY OUT ACTION
RECORD RESULT OF ACTION
RECORD STATE OF PARENT GOAL
LEARN ANY LESSON FROM EXPERIENCE
FAIL GOAL
PRUNE TREE BY GETTING RID OF ACHIEVED OR FAILED GOALS AND PLANS
NO
IS MAIN GOAL ACHIEVED OR FAILED ?
YES
REPORT RESULT

Figure 3.10a Achieve goal module (adapted from Power, 1974).

action or update its view of the world in the light of an action. The second module contains the main planning algorithm that essentially searches for a belief relevant to the goal[15]. It works as follows: (1) Input goal, initial conditions and state of the world. (2) Find a rule whose effect is the same as the current goal or sub-goal. (3) Check all preconditions are true. (4) If a precondition not true make it a sub-goal and repeat from step 2 or else continue. (5) Execute action. If action successful, set current goal or sub-goal as achieved. (6) Is main goal achieved? If yes finish, if no repeat from step 2.

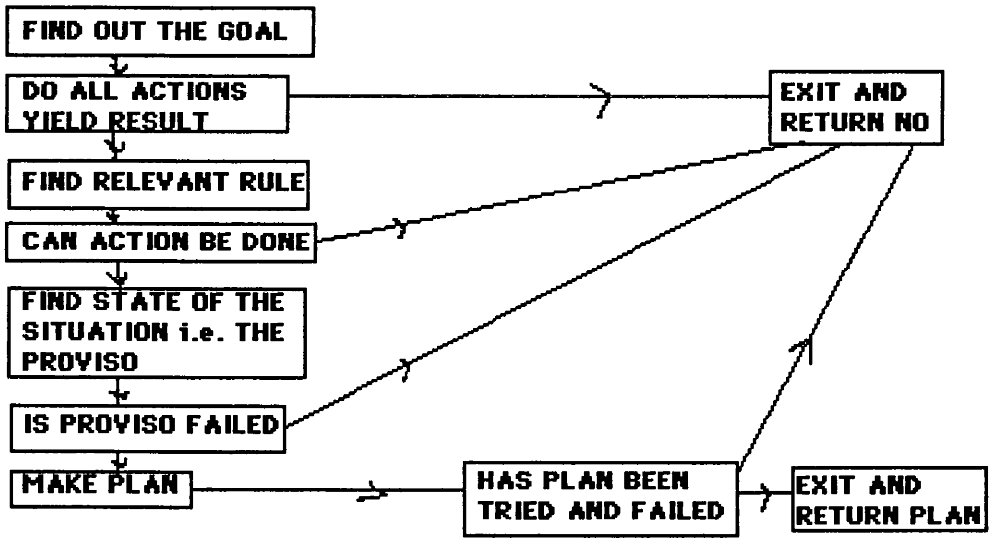


Figure 3.10b Planning module (adapted from Power, 1974).

Houghton's planner is in principle the same as that of Power. He refers to "Achieve goal" as the execution phase of planning. He overcomes circular plans by repeating the planning

[15] One weakness with this latter algorithm is that circular plans occur quite frequently since the beliefs do not specify which agent is to do the plan. See also the end of chapter 5 for a further discussion on circular plans. I draw the reader's attention to this because the problem of cycle detection occurs quite frequently in Artificial Intelligence problems. Dialogue systems must also give this special treatment since I am suggesting it is an inherent property of a large class of purposeful conversation. Sometimes it helps the main goal, sometimes it does not.

cycle twice; once for each agent assuming the plan failed first time around, as illustrated in detail in figure 3.10c. This is not, however, ideal for multi- as opposed to two-agent planning. It works, but a more appropriate solution seems to be to detect

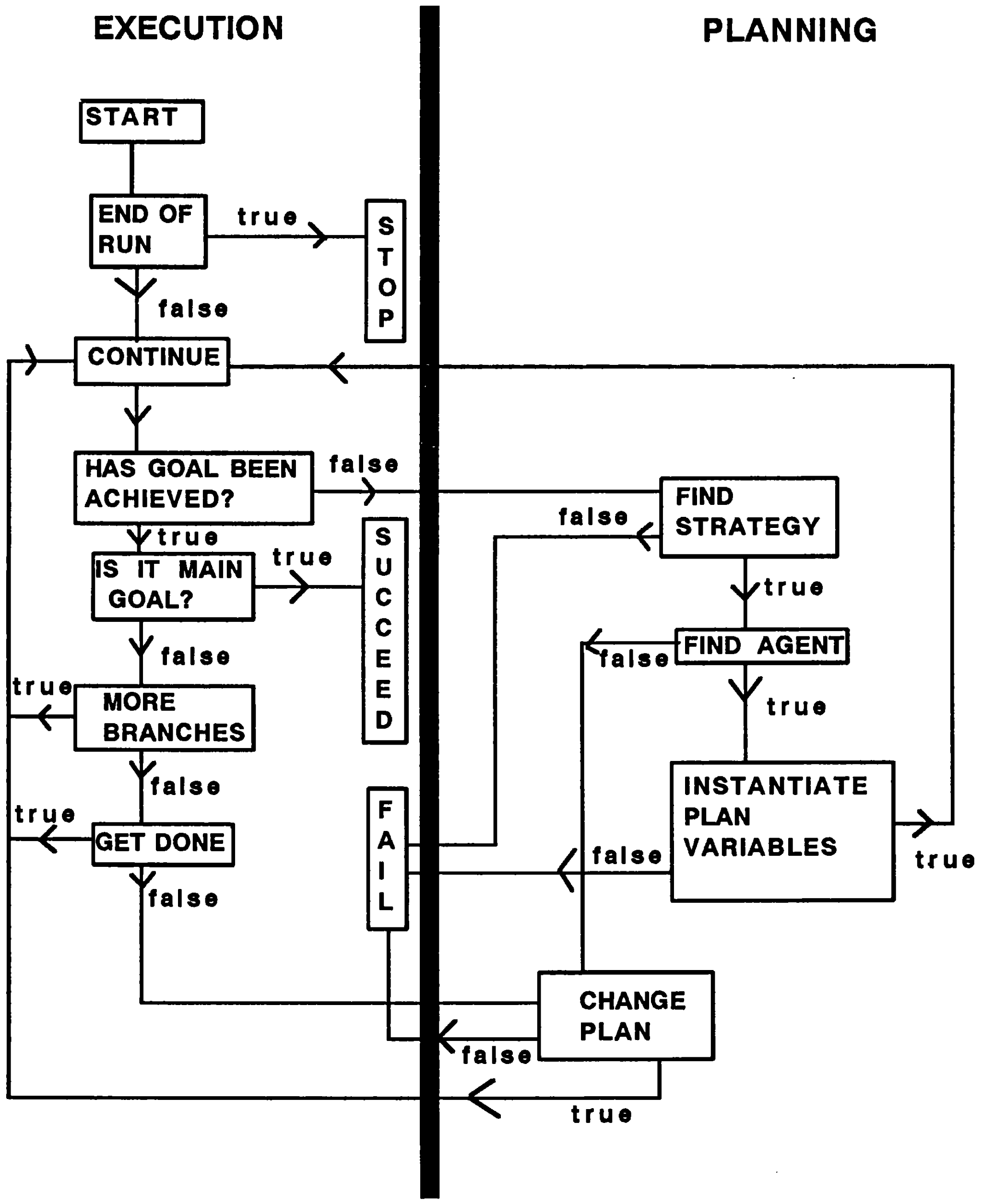


Figure 3.10c Planning module (adapted from Houghton 1986).

repetitions in the plan tree, instead of waiting for a failure, then to delete them and remember what to avoid the next time around. Another difference between Houghton and Power's systems is that Power's does not handle plan failures at all. For example, one type of plan failure occurs when the other agent already knows that part of the plan is true. In this situation Houghton's system automatically clips the plan tree and planning continues as though this was not part of the execution of the plan. In these situations, Power's algorithm exits from the planner and the agents then simply state that "we've got muddled". Houghton's system, like Power's, makes calls to conversational procedures within the planning module. Although I deal with this situation in more depth in the section on conversational procedures, it should be emphasised that the central problem with this and conversational planning is that, for Power, conversational procedures do not relate to goals. If a planner is trying to find something out, it makes an explicit call at the appropriate point within the planner. This is a reflection of the programmer's knowing what he would do in such circumstances, rather than the system's being able to work it out. Power's system makes calls to seven different conversational games at different points in both the planning and execution phases.

Houghton has one general function that makes access to a number of different types of information. Among these are (1) finding out about the state of the world, (2) finding out whether something is true, (3) asking or finding out whether somebody can do something, (4) asking how to do something, (5) getting somebody to do something, (6) checking whether a plan or strategy has not been done, (7) linking the preconditions for speaking with appropriate objects and (8) preparing to invoke an Interactional frame to find out, draw attention, get to do something or to make something known. Houghton's planner is a substantial improvement on Power because it deals with both when and how to produce an appropriate utterance whereas Power's only deals with when.

A theory might treat conversation and action in three different ways: independently (as in Houghton's system), dependently (as in Power's system) or identically (as in the "ideal" system). I take this latter method as a possible enhancement in chapter 6. The main point is that no conversational calls should be made during the planning and execution phase but instead at one specific point within the planner only. In order to do this, static (i.e. parameters specified at source) and dynamic (i.e. the joint plan tree and control stack that specify mutuality) knowledge representation needs to include both physical and conversational rules together and not separately. The planner should then be able to assess this knowledge representation at the same time and not using

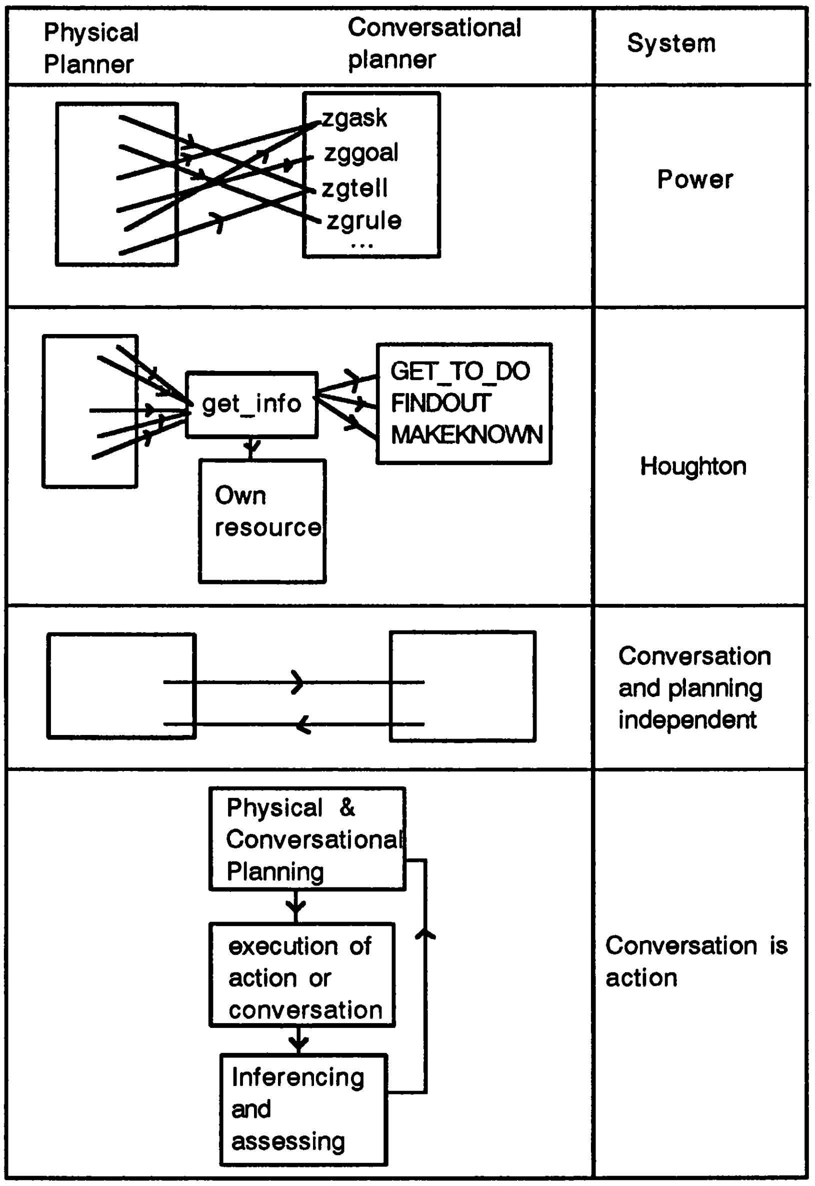


Figure 3.10d Diagram illustrating some of the alternative ways in which conversational and physical planning can be linked and perceived in relation to one another.

two different planners as implemented both by Houghton and

Power. Houghton's system is slightly better than Power's in that only one procedure (get_info) is used within the physical planner to search for what it wants.

Thus, the main weakness of these algorithms is that they both have no method for inference and they also call the conversational planner whenever they choose to. Ideally a call to the conversational planner should either be at exactly the same place or thought of as the same as planning (discussed more fully in chapter 6 and illustrated in figure 3.10d. The reason for this is that if such a system were to exist then it would be fully explicit in the sort of utterances it could make. It would then be possible to consider the problem of efficiency that was the concern of Houghton. If a system were not to follow this principle then problems of coordination and reasoning clearly infiltrate when for example the time slicing dictates an unexpected occurrence of either a physical or conversational event.

3.4.5 Conversational Planning.

In Power, the planning routines make calls to what he refers to as conversational procedures or games. They specify the form the conversation should take, what should be said and who should say it. There is an initiator and a respondent in each game, which are labelled as white and black. Once a game has been initiated there is a rigid set of rules that decides what gets said next. If one agent breaks a rule then the other interrupts with a suitable error message saying that a muddle has arisen and that “we'd best start again”. Before actually playing a game priming takes place as an agent announces his or her intention to play a game before the game itself takes place. So a normal human adjacency pair, such as “Is the bolt Up?”, "No”, would take four utterances due to the two initial lines of priming i.e. “May I ask you something?”,"By all means”,"Is the bolt Up?”, “No”. This allows both robots to know which game is taking place at any one time. Power defines seven conversational procedures. To take an example consider the procedure ZGASK. It has the following definition:

```
GAME ZGASK;
1. * W ZGQUERY [ZDQUERY];
 2. * B ZGANSWER [ZDSIGN];
3. ^ W ZGRECORD;
```

The numbers indicate the order in which things should take place. The symbols * and ^ indicate whether an entry or an utterance should be made. W and B stand for white and black and indicate who is to carry out the utterance. The main function ZGQUERY allows for a question to be asked, ZGANSWER allows for the response and ZGRECORD records what has been said in response. The functions in brackets determine how to interpret the utterance after it has been made.

In Houghton's system the implementation is the same as Power's except that, instead of using conversational games, interactional frames are used in conjunction with what is referred to as a POP11 process. This is an inbuilt process that allows the programmer to simulate parallelism. So that instead of using a control stack, as Power did, Houghton creates a series of sub-processes that can be suspended and reinstated when required. The advantage of Houghton's system is that the code is much easier to understand. The disadvantage is that agents may be required to reason about the process (e.g. when an unexpected event occurs), some subset of the process or the process content and therefore without knowledge of where the process is, or about parts of its content, this program representation is not entirely satisfactory. This is why I choose (and discuss in chapter 4) to continue with Power's control stack representation and avoid using processes.

In Houghton's system, interpreting utterances is ignored, there is no priming and, when an agent has finished speaking, the conversational structure is automatically passed to the other agent whose turn it is to reply. On generating an utterance each frame has additional rules which determine what has to be true before such an utterance can be made and what has to be done afterwards. Further details about this can be found in Cohen and Perrault (1979a,b). In Houghton, there are four interactional frames: MAKEKNOWN, FINDOUT, GET_TO_DO and GET_ATTENTION. I give here the definition of MAKEKNOWN:

```
INTERACTION_FORM
MAKEKNOWN
ROLES speaker=robot hearer=robot  prop=fact
GOALOF [know hearer prop]
MEANS  [know hearer [know speaker prop]]
PRECOND prec2 knows(speaker, [not [know ^hearer ^prop]]);
RESPONSE [hearer updateworldview speaker check_acceptance]
REPLY_TYPE MAKEKNOWN
```

In order for an interaction to occur the following steps need to take place (this is what Houghton refers to as conversational planning or planning an interaction). (i) find a suitable interaction for desired goal, (ii) bind variables to the interaction, (iii) make preconditions true, (iv) make sure we are not already engaged in a conversation, (v) add instance to stack, (vi) prepare message structure and (vii) pass to language generation. A similar sort of conversational planner exists for responding to a message.

The dilemma all conversational planners must face is that, on the one hand, there is a need to bind your partner within a predefined script while, on the other hand, to do so would not allow your partner freedom to think. The task of a good hearer is sometimes to think in parallel about both the speaker's and the hearer's needs (e.g. when the speaker is saying something that does not need to be listened to and sometimes to interchange between both the speaker's and the hearer's needs (e.g. when a change in the state of the world occurs but is unnoticed by the current speaker). A conversational procedure or an interactional frame as currently implemented by these systems prohibits this. There has been some discussion about alternatives to CP's and IF's. For example Musson and Shadbolt (1987) looked at how Power's model contrasted with Allen's in that Power used conversational procedures and Allen did not. Nevertheless some kind of conversational structure along these lines is inevitable since to avoid doing so would introduce serious pragmatic ambiguity and would make coordination and control difficult if the knowledge or conversational skills varied significantly.

3.4.6 Meta-planning.

There is very little evidence of meta-planning, (e.g. Wilkins 1984, Sacerdoti 1974, Stefik 1980,1981) in either Houghton's or Power's systems since they both only deal with one mutually agreed plan tree at a time. If however agents were required to keep their own internal plan trees distinct from the mutually agreed plan trees and both agents were required to think in parallel, the effect would be to enable them to deal with a variety of interruptions and conflicting plans. Agents would not interrupt an adjacency pair or conversational procedure but seizing the initiative should be allowed to occur between sets of adjacency pairs. To some extent this is discussed by Power (1987) and referred to as presumption and efficiency. It is for this reason that there is a need for a conversational and physical meta-planner that looks at the overall problem and then plans what conversation, plan or action to perform next. It is this module that should be concerned with efficiency in dialogue.

3.4.7 Executive Control.

Whereas Meta-planning is concerned with monitoring the course of the process or piece of program that is currently being executed so as to ensure that it proceeds in the right direction, the excutive provides the mechanism that determines the next most appropriate course of action.

For Power there is a single queue of goals that are either "situations" (which need planning) or "events" which just need execution. For Houghton there is no conscious organisation apart from the use of the built-in POP11 processes mentioned earlier.

In Power's system he has what he refers to as an executive. The executive interfaces between the two agents and represents the top level control of the overall dialogue. It is what Power (1974) referred to as the chairman.

In Power's program there are two sets of codes, one for each agent. The executive then provides the coordination between the two programs determining which one runs at any given time. By slicing the program into meaningful units the executive can transfer control from one program to the other whenever it chooses. However in this implementation, no considerations over and above swapping alternately were considered. That is to say, each agent could only think for the same fixed amount of time. Thus the executive could not allow

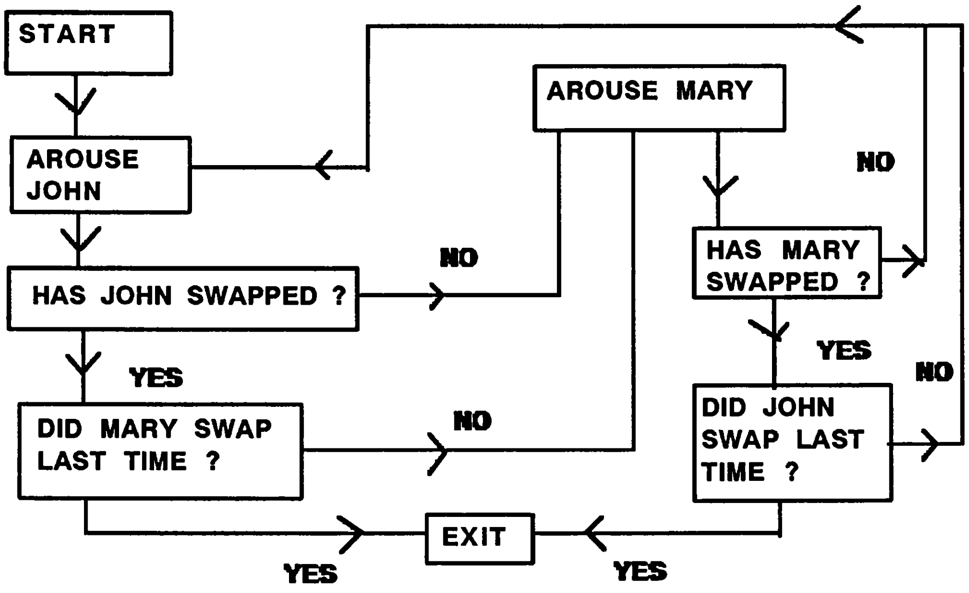


Figure 3.10e Executive Control (adapted from Power 1974).

for one of the agents to think faster than the other. Although this configuration is suitable for a metaplanner to reason over he made no claims about this.

In Houghton's system there is only one set of codes. Each agent is allowed to speak by constructing more than one process. In essence there is no chairman since one robot goes on thinking and speaking until he decides not to do so. This is achieved by suspending the current process at the point at which this is convenient. Immediately when a process is suspended the other agent resumes control of the program.

More general issues are relevant to conversation theory that the meta-planner and the central executive must concern themselves with. These issues include how to invoke a level of explicit conversation by the use of presumption and in turn how to interpret incoming events or conversation that involve a kind of interruption. This would involve a transfer of control between leading (i.e. by initiating CP's) and following (i.e. by having to respond to CP's all the time) the course of the dialogue. This latter concern forms the basis of the considerations looked at in chapter 4.

3.4.8 Initial setting and Representation.

In Power, typical knowledge representation at source (adapted from Power 1979) contains the following:

Positions of objects:
[JOHN OUT, MARY IN, DOOR SHUT, BOLT UP]
Perception:
John can see all four objects;
Mary is blind and cannot see any of them.
e.g. [door bolt robot]->jksee.
Action:
John can MOVE and SLIDE, but not PUSH;
Mary can do all three.
e.g. [move slide]->jkacts.
Goals:
John's goal is to be in.
Mary's goal is for John to be in.

Beliefs:
John believes that:
If a robot MOVES, nothing happens.
If a robot PUSHES the door,
the door changes position.
If a robot SLIDES the bolt, nothing happens.
Mary believes that:
If a robot MOVES, he changes position
provided the door is OPEN.
If a robot PUSHES the door,
the door changes position.
If a robot SLIDES the bolt, nothing happens.

In Houghton there is knowledge representation for a greater variety of uses. It is split into two major categories, "core" and "variable" knowledge. Examples of different types are as follows:

i) Core Knowledge reflects what the robots and the system know about the physical world. This knowledge never changes throughout the dialogue.

[Fred knows [opp X Y]][opp in out]
[Fred knows [sameloc isa position]]
[Fred knows [hand4 is hand]]
[Doris knows [hand4 partof Fred]]
[Fred knows [hand4 partof Fred]]
[hand4 partof Fred]]
[Fred knows [[door] objof move]]
[Doris knows [in isa position]]
[Doris knows [closed isa doorpos]]
[Fred knows [Fred isa robot]]
[Doris knows [Fred isa robot]]
[Fred isa robot]

ii) Variable knowledge is knowledge that changes during the course of an interaction. For example, if Doris slides bolt4 and tells Fred then a component of the variable knowledge changes to [Fred knows [bolt4 is down]].

```
[Fred knows [ Doris can push]]
[Fred knows [bolt4 is up]]
[Fred knows [ if [ robot push door]
              when [bolt up]
              then [door move]]]
[bolt4 is down][Doris knows [bolt4 is down]]
Fred knows [Doris can slide]
[Doris knows [Fred can movethru]]
[Fred is out][Doris knows [ Fred is out]]
[Doris is in] [door4 is closed]
```

For a goal of one of the agents getting In, Power has a maximum of 26 different knowledge representation entries. For the same domain Houghton has at least twice that many. Whereas Power has different variables to represent different types of knowledge, Houghton represents all knowledge in the form of [Fred knows [fact]] where "fact" is a list that contains a unique semantic identifier. In Power's case, the knowledge representation that Houghton uses is buried either in the program code or updated as another variable later on as in knowing what each other can do, for example.

In addition to this Power has no meta-conversational knowledge to be applied to conversation in terms of when it can be used. Instead it is directly called upon within the planner. On the other hand, Houghton has Cohen and Perrault's (1979a,b) speech act definitions for his IF's. Thus he has established the preconditions[16] for what has to be true before a conversation can take place.

There is also, in both systems, a dynamic knowledge representation in the form of a plan tree, which contains information about physical sub-goals that have a number of different plan varying facts (failed, notyet and achieved)[17]. Neither system considers knowledge that has a different status. Knowledge can come about from different sources, it can arise through the conversation, it can be told, inferred, estimated, known, unknown or assumed. For example, knowledge that has just been given (told) to an agent is not always true after an action takes place. Another example would be when you need to keep track of these time varying facts to avoid asking the same questions. Thus knowledge needs to be represented in such a way as to capture this dynamic aspect otherwise the information becomes out of date, as it would do if implemented

[16] It is not easy to identify clearly all the preconditions for speaking in spite of the attempts made by Cohen and Perrault (1979a,b) since this depends on the conversational and physical goals that are brought to the dialogue at outset. Nevertheless for a limited domain such as Power's robot world, specifying the preconditions is relatively easy. Amongst the most important that I have identified more recently are (1.) Speaker does not know that the hearer can not do or see it. (2.) Speaker knows hearer is being cooperative. (3.) Mode of interaction is joint (4.) You have not asked it before. (5.) You can not do it yourself. (6.) You do not know it yourself. (7.) You are not planning to ask already. (8.) You have not just inferred it. (9.) Are you being efficient?

Coates (1990) also makes the important point that in human dialogue sometimes the underlying intention of an exchange is only revealed after the utterances of the speakers are complete. She backs this argument up by giving an example dialogue in which two speakers are constructing the same utterance together. However even this example is also not entirely satisfactory since both speakers were speaking simultaneously in a dogmatic manner. Thus although the intention is always unclear at any particular junction in time, the tone often is, e.g. husband says to wife " I am going to be late tonight darling" is only understood by how it sounds.

[17] More recent systems such as Carletta (1990) have included the following categories (open, unplanned, unexecuted and executed).

using Power and Houghton's representations. Thus in short Houghton and Power both allow their knowledge at source to undergo physical changes. Houghton's is superior to Power's in that the representation is systematised in one database[18] . Both have limitations in that information about the operational status of knowledge is not retained. By expanding the variety of knowledge skills of the agents, SUPERPOWER is forced to confront this problem in chapter 5.

3.4.9 Conflict Resolution[19].

For the successful execution of dialogue it is vital for partners to establish and maintain agreement about all items and have ways to agree them. in order to achieve this aim agents must possess the ability to detect conflict and then resolve it. There are a number of instances when this problem occurs in Power and Houghton.

In some cases conflict arises in response to suggesting a plan of action. Instead of saying "I can't think of one", as Power

[18] In POP11, there is a facility that allows for data to be stored in one variable. There are then library functions that allow the user to access the database in different ways for example by pattern matching.

[19] There has been some work, by Galliers (1987) using modal logic that discusses the process of trying to either co-operate or resolve conflict in dialogue. This work concluded that conflict is as co-operative and perhaps even more so than for example benevolence, since the conflicting negotiation serves to move agents towards a mutually desired goal. These are certainly the findings of this thesis; in some cases the agents physically appear to be entering dialogue for long periods of time in which they are both moving further from the main goal. For further discussions on the nature of mutuality, see for example Bratman(1987), Searle (1990), Cohen and Levesque (1990) and Power (1984).

does, Houghton comes back with the question "How do you do X ?". However neither system can deal with the case when neither agent can develop an appropriate plan.

After constructing a joint plan, discussions may arise if the plan is inconsistent with a belief. Both Houghton and Power can deal with this in the same way. When humans discuss what is right or wrong in terms of their beliefs about the world there is a situation in which they have to make up their minds about which belief to adopt. In some cases the one that wins is the rule or belief that offers more specific knowledge. In other cases it is the rule that "sounds" the better. For Power and Houghton, agreement and disagreement about beliefs is

1.	John:	I want to explain something
2.	Mary:	Go ahead.
3.	John:	If you move nothing happens
4.	Mary:	I disagree.
5.	John:	--.
6.	Mary:	If you move when the door is open you change position.
7.	John:	I see.

Figure 3.11a A discussion about a belief in which the more complicated rule is judged to be the better one.

resolved, usually with the longer rule taking priority. That is to say, a rule that does not contain a precondition is rejected in favour of the one that does. Both systems do this. An example, taken from Power's system is given in Figure 3.11a above.

After an action takes place there is a need within the planner to infer exactly what has happened in order to resolve any conflict that might arise from the consequences of the

1.	John:	Let's assess the consequences of my action.
2.	Mary:	Go ahead.
3.	John:	Nothing has happened.
4.	Mary:	I disagree.
5.	John:	- - .
6.	Mary:	The bolt has changed position.
7.	John:	I see.
8.	Mary:	The bolt is now up.
9.	John:	Right.

Figure 3.11b Action failure in which one of the agents believes that an action has not achieved its intended goal.

new state of the world. In Power's model, an action failure is handled within a conversational procedure that is automatically invoked after the action has taken place, as illustrated in figure 3.11b. Houghton does not consider dialogue after an action has taken place.

There are other more open-ended senses of conflict too. Neither system discuss the more common situation in which both agents are in conflict in the sense that they both know that they are confused about how to achieve the main goal. As a corollary to this problem agents may also be confused about their knowledge or the current state of the world. Some solutions to this problem are the concern of SUPERPOWER in chapter 5 and are mentioned later in this chapter.

3.4.10 Interaction skills.

Interaction concerns itself not with the preoccupations of individual speakers or hearers, nor with a sensible plan to achieve an overall task but with "evidence for interplay between participants" (Anderson, Clark and Mullin 1989). Other important features of interaction addressed here include freedom to seize control and the business of CP's and IF's being kept open and at the same time unachieved. In this section, I look at these systems' abilities to describe the agents' interchange which is a major focus for SUPERPOWER.

In Power's system, the first place where interactional skills can be seen is in turn-taking, with no repetitions of the same type of CP at any one time. Only one plan (or several sub-plans) can be active at any one time. Thus interaction is best described as short and sharp with at most two or three conversational procedures active at any one time. There is no level of dominance and the robots are always thinking about the same plan or conversational procedure one after the other. Co-operation is achieved through slavery on the part of one or other of the agents and the ability to interact is restrained to the reasoning of the conversational procedure itself.

Coordination is achieved by sending one of the agents to "sleep" until the other has caught up to the same point within the program code. Interruptions are hard-coded into the planner at appropriate points for naturalness as opposed to some necessary interruption reasoning process that takes into consideration the ramifications of the other participant. Interruptions are also modelled in Power when some tacit disagreement hinders advance towards the goal actively and an interjection is appropriate, for example, the use of "call by name" as in "1. John: Mary, 2. Mary: Yes". There is also some inferencing after an action takes place that is automatically and immediately hardwired and executed.

For Houghton, interaction is turn-taking and any number of interactional frames can be invoked or active at any one time. Thus conversation can be embedded so that a number of interactional frames are active at any one time. Just as with Power, it is not clear to what extent conversation that occurs before an action can be retained active after that action. Thus although the interaction frames are clearly embedded at any number of levels, the assumption is that there has been no

1. John:	How do you get to be in?
2. Mary:	You open the door and move.

<u>Figure 3.11c Being told how to do something (adapted from Houghton 1986).</u>

change in the current goal. Thus physical planning is the same as for Power. So are dominance, co-operation and coordination. However reasoning about plans and different bits of knowledge can vary considerably. For example, in Figure 3.11c John is thinking about how to achieve the goal of being In whereas Mary is planning how to be in.

In the next two chapters, I show that a critical milestone for any dialogue system is the ability to interact even when there is a change in the state of the world. This involves some improved knowledge representation that reflects new and old knowledge that retains its validity and a recursive algorithm for keeping track of conversational procedures that are kept active for long periods of time over perhaps several actions and are then needed or completed at a later time.

3.4.11 Inference.

In Power's system all inference is hardwired within the conversational procedure. For example, in figure 3.11d, the following exchange all occurs within one conversational procedure.

1.	John:	Let's assess the consequences of my action.
2.	Mary:	By all means.
3.	John:	The door is open.
4.	Mary:	I see.

Figure 3.11d Action success (adapted from Power 1974).

In Houghton there is no need for inference since the planner is always good enough (in relation to achieving the main goal) to avoid this. Later, in SUPERPOWER, all inference is done exhaustively within the planner. Thus, with Houghton, once all plans are negotiated it is assumed that the actions will achieve their desired effect. But with Power's system an actor engages in a conversational procedure "zgassess" immediately after it performs an action with the intention of making sure that both agents understand what has happened. On completion, both agents exit from the game, update their plan structure and infer a new belief about the world if the current change was not what was expected.

One problem that perhaps has often been underestimated is that modularising a program often restricts its scope for cognitive development. For example, Houghton (1989) modularised utterance planning such that the syntactic structure of an utterance was determined by a plan that could be either physical or conversational. Then the utterance could be generated using Halliday's systemic grammar. It is not at all clear whether this took into consideration the circumstances from which the plan was derived in the first place. For example, how could it distinguish between "The bolt is Up" and "The bolt must be Up"? The former might have been an answer to a question and the latter an inference, yet the meaning are the same. These two sentences are neither given nor new facts but instead have their grammatical structure rooted within the dynamic planning process of inferring something.[20] Thus, although systemic grammar readily distinguishes between these two modalities and Davey (1978) applies this in his system it has not been extensively applied in this particular instance. The problem of relating the dynamic aspects of planning to modality in linguistic structure (see also Coates 1990, Palmer 1990 and Stubbs 1987) is an important milestone for inference since once the variety of inference increases a respondent will need cues to understand how an utterance, containing an inference, was actually made.

I hope to be showing in future chapters that SUPERPOWER's inference capabilities combined with the planning mechanisms permit solid support for a more general account of modality.

3.5SUPERPOWER.

I now focus on the two main differences between my system and those of Power and Houghton. First to be considered will be the coordination i.e what Power refers to as the

[20] A useful modular account of computational linguistics for reference purposes can be found within Gazdar and Mellish (1990).

executive control in terms of when the agents are allowed to think. This is the component where any meta-planning that needs to be done should be done. It is the starting point for allowing agents to vary the dialogue interaction as a whole.

In terms of coordination, the agents must be able to interrupt one another whenever they want to. In Power's system, this is only possible when it is the appropriate agent's turn. In Houghton processes are used, but again, only when one agent has finished speaking is this possible. In SUPERPOWER both agents are allowed to think at the same time. A comparative summary is shown in figure 3.12 which indicates a short bar line when it is the appropriate agent's turn. Only one of the agents is actually being allowed to think. In the case of humans, perfect parallelism can exist, although it is normally very difficult to talk when you are trying to listen at the same time, likewise it is very difficult to think about one's own goals at the same time as listening. Thus with humans, to be part of an interaction participants must accept as a minimum these constraints. Apart from this, they are free to talk or listen as and when they choose. In Houghton and Power's systems there are periods in which both agents can have absolutely no impact on the dialogue. In SUPERPOWER, I have allowed the thinking time to be distributed between the two agents allowing them the freedom to think, talk and listen

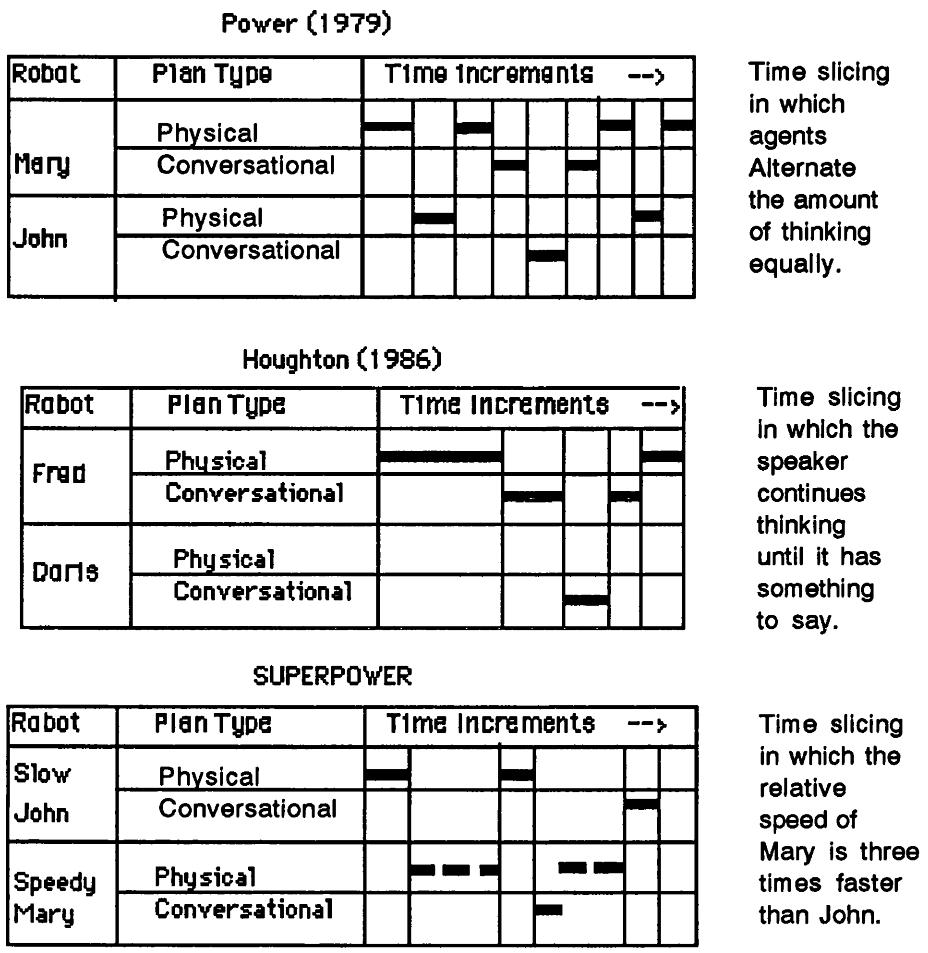


Figure 3.12 Time slicing illustrating the degree of parallelism that can exist between one conversational simulation and another. In Power, control swaps alternately. In Houghton, control changes when one agent has reached a logical conclusion. In SUPERPOWER one agent can be allocated more time increments per swap than the other. Thus, with this allocation, the robots are thinking at the same time because such a simulation could not possibly work unless the program code had been designed so that each robot thought independently of one another.

more independently than before but observing the human principles of not planning and listening or talking and listening at the same time.

Secondly, SUPERPOWER has improvements in terms of the depth of planning and inference techniques that are possible. Figures 1.10 (chapter 1), 1.9 (chapter 1), 3.13 and 3.14 illustrate the main types of dialogue that are addressed. In the first instance there is classical planning, figure 1.10 (chapter 1), which is the type of algorithm that both Houghton and Power use. Whereas Instrumental planning relates to the need to overcome perceptual difficulties, experimental planning relates to the need to overcome not knowing how to do something.

Within experimental planning Power's program simply

		(State of the world is now [Mary in, John out, door open, bolt up])
1.	Mary:	Is the door open?
2.	John:	I don't know.
3.	Mary:	Let's do a test. I suggest I move.
4.	John:	All right.
		(State of the world is now [Mary out, John out, door open, bolt up])
5.	Mary:	I have moved.
6.	John:	I see.
7.	Mary:	I have changed position.
8.	John:	I see.
9.	Mary:	I am now out.
10.	John:	Right.
11.	Mary:	The door must be open.
12.	John:	Right.

Figure 3.13 Instrumental Planning in which not knowing something generates the need for both conversation and action.

aborts. In Houghton's program he considers experimental planning but only when a single agent does not know how to plan. In the dialogue illustrated in figure 1.9 (chapter 1) I demonstrate a new set of dialogue possibilities by generating sentences appropriate for when both agents cannot find a plan. Then there is instrumental planning (figure 3.13) in which neither agent can perceive the current state of an object in the world and they need to work out how to discover what it is. Neither Houghton nor Power consider this or the other case which involves dialogue about what to do when something

		(State of the world is [door shut, Mary in, John in, bolt down])
1.	Mary:	Will you help me get out?
2.	John:	By all means.
		(State of the world is [door shut, Mary out, John out, bolt down])
3.	Mary:	I have changed position.
4.	John:	I see.
5.	Mary:	I am now out.
6.	John:	I see.
7.	Mary:	Somebody has put me out.
8.	John:	Really.
9.	John:	I have changed position.
10.	Mary:	I see.
11.	John:	I am now out.
12.	Mary:	I see.
13.	John:	Somebody has put me out.
14.	Mary:	Really.

Figure 3.14 Unexpected Planning in which both agents have agreed a goal that is suddenly realised automatically. The task then becomes a matter of inferring what has happened.

unexpected occurs. In figure 3.14, both agents are in dialogue with one another when suddenly they are both placed in the Out position. The task is then for both agents to assess and infer exactly what has happened. This falls within the definition of

Unexpected planning[21].

3.6 Summary.

In this chapter, I looked first at early dialogue systems that do not consider purposeful conversation in terms of any theory. Next, I discussed goal-directed systems that do provide such a theory and some work that went into understanding the components of a goal-directed system. I explained that one area not covered by this work is the ability of the agents to enhance their pragmatic skills as a result of their having been endowed with different knowledge, perception, action and speed skills. I then went on to discuss Power's and Houghton's systems that come the closest to achieving this aim.

In chapter 4, I shall discuss the time slicing algorithm that allows agents to think and converse whenever they choose. I shall also explain how speed changes and differences in other skills give rise to different dialogue. Then in chapter 5, I shall discuss an exhaustive joint planning algorithm for a wider variety of skills than that reported so far.

[21] Unexpected planning has been the interest of Social Scientists recently. For example, Suchman (1987) suggests that the work in Artificial Intelligence planning has little to say about how humans perform tasks. By illustrating her theory with dialogue about photocopying she shows how plans are not developed in advance of the action but instead as a consequence of the situation. However by simply being familiar with a type of event and inferring what is happening from past experience robots can also develop plans "*in situ*". Agre (1990) also supports this argument.

Chapter 4.

Time slicing.

4.0 Introduction.

In the previous chapter I identified two problems that other dialogue systems have not considered namely the time slicing and kinds of planning. In this chapter I discuss the first of these problems.

Power (1974), both at the end of his thesis and in more recent articles, Indicates a number of problems that his original program failed to address. A better representation than that first described in Power (1974) could be achieved by swapping the control from one agent to another at all stages of planning, thus allowing simulation in parallel. This is what I refer to as time slicing. Power (1979) also suggests it would be more satisfactory to have a control stack representing the point of the utterance rather than a mixture of information about mutuality, utterances, dialogue, and joint plans and (Power, 1987) discusses the need for the dialogue to be more efficient. Finally, Power (1984); (see also Power and Dal Martello 1986) recognises that the problem of what constitutes agreement still needs to be resolved. Some recent programs have partially solved some of these problems but not all.

It is with these ideas in mind that this chapter now takes up the issue of time slicing and explains how different timing accounts for variety in dialogue. I will describe the computer simulation of Power's robot world. The program listing in the appendix is an enhanced version of the program described in Power (1974,1979) and represents dialogue generated by SUPERPOWER V 5.9, described in this and the next chapter. Almost all of the original variable names, procedures and programming conventions that are described in technical detail in Power (1974) have been retained.

In section 4.1, I discuss the general problem of reasoning about events and processes which is at the heart of time slicing. An event may either be independent of, or dependent on, the current process[22]. Put simply, does it interrupt something or is it exactly what an agent was expecting? The main choices are listed in figure 4.1. Although the agents in SUPERPOWER do not actually reason deeply about this, there are mechanisms to access any point within a process provided it

1) An event occurs and the agent was not expecting one.
2) An event occurs and the agent was expecting one.
3) An event does not occur and the agent was expecting one.
4) An event does not occur and the agent was not expecting one.

Figure 4.1 The general problem of processing events (utterances or actions). This provides the mechanism for what to do when something unexpected happens, thus improving the robustness of interaction.

has a reasonable amount of functional significance. Simple actions such as deleting, advancing, suspending or continuing a

[22] Lansky (1986,1988), Georgeff (1987) have taken this issue up theoretically in GEMPLAN for both single and multi-agent domains but not, however, for problems relating to dialogue.

process, once the executive control has interpreted how incoming events relate to existing processes, are included. Thus, the general problem of reasoning about an event is dealt with in section 4.1.1 followed by a discussion of when this occurs as well as how to reason about the current process (section 4.1.2). Sample dialogues are illustrated to show what happens if the agents are allowed to think independently in section 4.1.3. Next, in section 4.1.3.1 and 4.1.3.2 I detail a Monte Carlo simulation by considering all possible combinations of agent skills, with speed changes, and show how it is possible for them to give rise to changes in dialogue.

In the last section (4.2), I explain how these improvements are important for organising the dialogue according to the four levels discussed in chapter 2. Level 4 is about occupying the airwaves and so the state of expecting a reply (section 4.1.1) helps to decide what to do when an event or utterance occurs. Relative speeds are important at this level, since, when an event occurs that is independent of internal processing, it provides a trigger to stop planning and pay attention to the interrupt. I have not modelled how to reason about an interruption but have provided a program structure that could enable this (e.g. figure 4.3).

4.1 Allowing both agents to think in parallel - an overview.

A closer simulation of independent robots, processing in parallel, can be implemented by expanding the central executive (Power 1974: exec) in such a way as to swap control incrementally from one agent to another. Processing time is split into small chunks of planning and conversational steps. Control is not now passed only when an utterance is produced, as in the earlier models of Houghton (1986) and Power (1974), but instead after a number of predefined chunks of reasoning determined by a robot's speed parameter. This tests conversational coordination mechanisms of the agents - there are now many more moments when the other may speak. They can be "surprised" and "interrupted". This brings out the theoretical problem of how an agent should respond to an event (utterance), whether or not it is expected.

In order to achieve all this, a program simulation must be able to process both existing and input activity at the same time. This in turn involves relating incoming events (physical or conversational) to current processes (physical plans or topics). There is a variety of strategies for dealing with this, depending on the style of agent that is required. For example, following Power (1979), one strategy is hardwired into the program in order to be able to ignore current processing, to respond immediately and to engage actively when an agent is spoken to. He refers to this as call by name (Mary says: "John", John replies: "Yes" and suspends his current internal processing). Situations often arise where this strategy might not be appropriate, and in this case the agents would have to know how to reason at this level. I do not consider this problem here but instead expand the notion of expecting a reply in the central executive of the agents.

Next, I introduce the reader to the problem of defining an increment of time that can be used as a method of simulating parallel activity. Later on I shall be showing that one of the most striking conclusions from this is that what gets said in a dialogue is dependent on the relative speeds of the participants, even when all other initial conditions remain constant. This is surprising, since one might expect only the order of dialogue to be affected. Furthermore, in some instances, even the outcome of the dialogue can be determined by the relative speeds of the agents.

4.1.1 The notion of expecting a reply.

In human dialogue the hearer is always confronted with the problem of how to balance resources between listening to

Given that:

1) An event has just occurred and the agent was not expecting it. Then:
e.g. 1. The agent could have been finishing off interpreting another message.
e.g. 2. The agent could have been thinking about something else.
e.g. 3. The agent could have been preparing to say something.

2) An event has just occurred and the agent was expecting it. Then:
e.g. 1. The agent could have been planning, assessing or saying something else.
e.g. 2. The agent might simply be waiting for the event.

3) No new event occurs but the agent is expecting one. Then:
e.g. 1. The agent could have been waiting for the other to say something.
e.g. 2. The agent could have been waiting for the other to perform an action.

4) No new event occurs but the agent is not expecting one. Then:
e.g. 1. The agent could have been interpreting a previous utterance.
e.g. 2. The agent could have been preparing to make an utterance.
e.g. 3. The agent could have been planning to make an utterance.
e.g. 4. The agent could have been preparing to make an utterance but was still planning or assessing.

Figure 4.2 The notion of expecting a reply illustrating the main processes that an agent is involved in at the time of an event (or non-event). Both the process and the event require reasoning about.

the speaker and preparing to speak himself. In SUPERPOWER, I pave the way for this problem by introducing the notion of expecting a reply. Furthermore, all processes within SUPERPOWER are subdivided into functional units that can be suspended at any time. Thus, if an event should warrant it, a new process can start and when it is finished the old process can be resumed. There is nothing new in this idea except that in SUPERPOWER the dialogue can be more explicit. The program always recognises the type of event that has just occurred and therefore responds accordingly. The various cases that an agent's executive (Power 1979) must consider, when control is

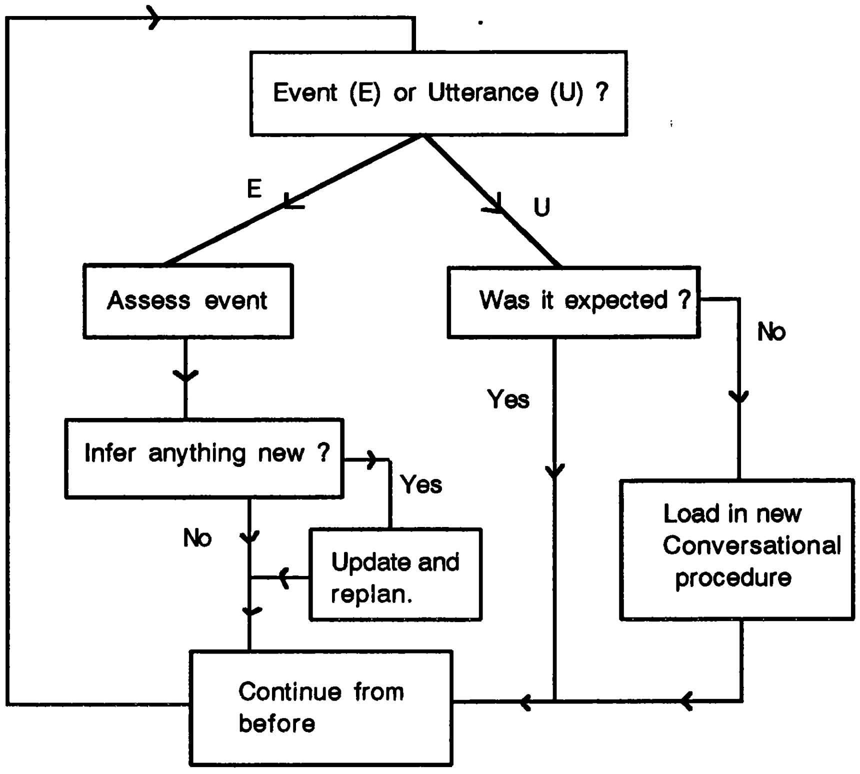


Figure 4.3 How incoming events and utterances are processed when robots are allowed to think in parallel.

handed back to it, are illustrated in figures 4.2 and 4.3 above.

4.1.2 An increment of time.

The problem of simulating the processes of two agents in parallel is not obvious. Processes must be simulated in such a way as to alternate the amount of CPU time each agent's conversational program can occupy. This in turn means that each agent's process must be allocated a certain amount of the CPU time each time the simulator's chairman hands over control to an agent's program. This then poses the problem of how to define how much of this processing represents one time unit? On the one hand, the increment should not be too small since conversation has very little to do with computer machine code and on the other it should not be too large since it would not test the flexibility of the mechanisms very much.

Thus, at one extreme every machine code instruction could count as an increment. At the other is Houghton's (1986) definition based on control being passed only when agents have finished a process of making an utterance. A third definition (Power 1979) could be based on the idea that every instantiation of a plan step or a conversational procedure constitutes one increment. An even smaller increment might be useful in certain situations (e.g. Mary might be looking up some knowledge while John tells her what she is trying to find out). SUPERPOWER uses Power's (1979) definition but is designed so that any definition would be easy to implement.

Although there is considerable debate as to what this could actually mean, an increment of time in conversation is nevertheless usefully defined when, on execution, a segment of a conversational program takes on a meaning that can be reasoned about. This provides the basis for a variety of situations which would otherwise not be handled by dialogue systems. For example, a difficult situation could arise if one stupid agent is very much faster than a clever but slow agent. The question arises of whether or not to remain within the conversation dictated by the stupid but fast agent. Although more related issues such as pauses, hesitations or interruptions (Butterworth 1980) are not directly considered here, a clever agent would not acquiesce but pause and think. Similarly, human conversation becomes more efficient as a consequence of reasoning about chunks of earlier discussions. Thus an important feature of any computer program that generates dialogue is for it to be structured in such a way that each conversationally meaningful function is segmented as a time unit that can be reasoned about. For example, if one unit related to responding to an utterance it means that the functional unit is conversational rather than physical or mental.

At the heart of all computer models of conversation there is a problem. Within the interactionist approaches such as Shadbolt and Carletta there is no allowance for speed disparities or differing knowledge skills. These are models of conversation only between identically-trained conversants. On the other hand, models that have an element of disparity between the speaker and the hearer by way of an interaction frame or a conversational procedure suffer from other kinds of defects. Namely, once an agent is in a CP or IF it has no way of getting out and is forced to respond in a direct way.

Part of this same problem is the dilemma between an elliptical and an explicit approach to modelling dialogue. An interactionist model is elliptical since it strives to specify all the important elements, such as Shadbolt's communicative postures, of an efficient dialogue system without exhaustive illustration of any conversational technique. My explicit approach with SUPERPOWER is to strive for fewer initial settings but greater amounts of explicitness for how a setting such as time slicing can be adequately tested through the use of a Monte Carlo simulation.

In the interactionist models such as Shadbolt and Musson (1987) there are no conversational procedures at all. The model relies on a form of constructive interaction. The job of each planner is to contribute to the previous utterance which is expressed in the form of a plan. Also it is not clear how they distinguish between listening and speaking at all. Thus it is a form of monologue, not dialogue, in this respect. The drawback with this method is that the planners cannot act differently, in the conversational sense, from one another. Agents can contribute provided their knowledge is similar. If, however, the knowledge of agents were varied, this method would not fulfil a sense of explicitness (since varied knowledege inevitably means a greater diversity in dialogue). By contrast, in models such as those of Power and Houghton conversational planning is treated independently from physical planning. In the former situation (i.e. Shadbolt and Musson 1987), the thinking processes associated with speaking are assumed to be the same. In the latter case, (i.e. Power 1979; Houghton 1986; Carletta 1990), a clear distinction is made between a planner that listens, by way of being bound to a conversational procedure or interactional frame, and a planner that prepares to speak. The advantage of binding within a CP (i.e. no control can be passed to a different point in the program once in a CP) is that time slicing becomes possible whereas without a CP or IF time slicing is impossible and therefore control or coordination cannot be illustrated within dialogue.[23]

4.1.3 Example dialogues with parallel processing.

I now demonstrate, through the use of SUPERPOWER what happens when agents are allowed to process independently of one another. Intuitively, the only difference would be the order in which things were said. This is not the case at all. Unexpected phenomena occur within dialogue when all factors

[23] One further important step forward is to minimise the amount of processing time spent within a CP or IF, and then to force this minimal procedure to behave just like a joint action. In other words there is a planner that reasons about conversation in the same way that it reasons about action. It also executes conversation rather like a joint action. Also within the program code, action is physically performed in the same place as the conversation (unlike Power and Houghton in which action and conversation are intermingled throughout the program code). Furthermore, for every physical plan of achieving a goal there is also a corresponding conversational method. Efficiency or ellipsis can then be improved by developing a theory of the hearer that knows when to interrupt combined with a theory about presumption, (Power 1987: 10), that assists the speaker in making elisions. This has the effect of allowing an infinite amount of explicitness that can then and only then be made efficient by developing conversational rules for situations such as a thoughtful hearer dealing with an uneducated, precipitous speaker. Minimising the amount of agreeing that occurs within a CP or IF is the main concern (discussed for future research in chapter 6) but, for the purpose of this section, part of the process of being more explicit means that each plan step, such as checking that preconditions are satisfied, must expose itself to be reasoned about. There are no short cuts, which is why in SUPERPOWER every process and joint plan is broken down into functional units so that more explicit conversation, if ever the need should arise, can appropriately be invoked. The listener then has two choices either to respond within the C.P. or metaplan his/her way out of it. This will be the concern of chapter 6.

```
*** what  John's capabilities are ***
1.  Goal: Help me get the door open.
2.  Position of objects:
    door: shut: seen,  John: out: seen,  Mary: undef: undef,  bolt: undef: undef
3.  Consequences of events:
    push: If you push the door when the bolt is up, the door changes position.
    slide: If you slide the bolt when you are in, the bolt changes position.
    move: If you move when the door is open, you change position.
4.  Range of actions:
    move: Yes,  slide: Yes,  push: No,

*** what  Mary's capabilities are ***
1.  Goal: None
2.  Position of objects:
    door: shut: seen,  John: undef: undef,  Mary: in: seen,  bolt: up: seen
3.  Consequences of events:
    push: If you push the door, the door changes position.
    slide: If you slide the bolt, the bolt changes position.
    move: If you move when the door is open, you change position.
4.  Range of actions:
    move: Yes,  slide: No,  push: Yes,

             Dialogue A                                   Dialogue B
             parallel processing set                      parallel processing set
             John's speed = 1                             John's speed = 3
             Mary's speed = 3                             Mary's speed = 1

1.  John:    Mary.                              1.  John:  Mary.
2.  Mary:    Yes.                               2.  Mary:  Yes.
3.  John:    I want to suggest a goal.          3.  John:  I want to suggest a goal.
4.  Mary:    Go ahead.                          4.  Mary:  Go ahead.
5.  John:    Will you help me get the door open? 5. John:  Will you help me get the door open?
6.  Mary:    By all means.                      6.  Mary:  By all means.
7.  Mary:    Shall we make a plan?              7.  John:  Shall we make a plan?
8.  John:    OK.                                8.  Mary:  OK.
9.  Mary:    I suggest that I push the door.    9.  John:  Mary.
10. John:    Mary.                              10. Mary:  Yes.
11. Mary:    Yes.                               11. John:  May I ask you something?
12. John:    May I ask you something?           12. Mary:  Go ahead.
13. Mary:    Go ahead.                          13. John:  Can you push the door?
14. John:    Is the bolt up?                    14. Mary:  Yes.
15. Mary:    Yes.                               15. John:  Mary.
16. John:    All right.                         16. Mary:  Yes.
                                                17. John:  May I ask you something?
 (State of the world is now                     18. Mary:  Go ahead.
 [John out, Mary in, bolt up, door open])       19. John:  Is the bolt up?
                                                20. Mary:  Yes.
                                                21. John:  I suggest that you push the door.
                                                22. Mary:  All right.

                                                (State of the world is now
                                                [John out, Mary in, bolt up, door open])
```

Figure 4.4 Relative speeds of the agents determining (a) utterance order, (b) utterance content (e.g. dialogue A line 9 and dialogue B line 21), (c) dialogue length, (d) dialogue content (e.g. dialogue B lines 11-14 does not appear in dialogue A), (e) interruption rate (i.e. dialogue A lines 7 and10 are interruptions whereas dialogue B has none).

remain the same except the relative speeds of the agents.

Figure 4.4 illustrates contrasting discussions in which the speeds of thought, which are measured in increments of program code (Sect 4.1.2 p103), of the agent in the left dialogue are reversed (all other factors remaining constant). It clearly

Dialogue A
parallel processing set
John's speed = 1 Mary's speed = 3

1. John: Mary.
2. Mary: Yes.
3. John: I want to suggest a goal.
4. Mary: Go ahead.
5. John: Will you help me get the door open?
6. Mary: By all means.
7. Mary: Shall we make a plan?
8. John: OK.
9. Mary: I suggest that I push the door.
10. John: Mary.
11. Mary: Yes.
12. John: May I ask you something?
13. Mary: Go ahead.
14. John: Is the bolt up?
15. Mary: No.
16. John: Mary.
17. Mary: Yes.
18.. John: I want to explain something.
19. Mary: Go ahead.
20. John: If you push the door when the bolt is up, the door changes position.
21. Mary: I see.
22. Mary: I suggest that we get the bolt up and then I push the door.
23. John: All right.
24. Mary: Shall we make a plan?
25. John: OK.
26. Mary: I suggest that I slide the bolt.
27 John: All right.

Dialogue B
parallel processing set
John's speed = 3 Mary's speed = 1

1. John: Mary.
2. Mary: Yes.
3. John: I want to suggest a goal.
4. Mary: Go ahead.
5. John: Will you help me get the door open?
6. Mary: By all means.
7. John: Shall we make a plan?
8. Mary: OK.
9. John: Mary.
10. Mary: Yes.
11. John: May I ask you something?
12. Mary: Go ahead.
13. John: Can you push the door?
14. Mary: Yes.
15. John: Mary.
16. Mary: Yes.
17. John: May I ask you something?
18. Mary: Go ahead.
19. John: Is the bolt up?
20. Mary: No.
21. John: I suggest that we get the bolt up and then you push the door.
22. Mary: All right.
23. John: Shall we make a plan?
24. Mary: OK.
25. John: Mary.
26. Mary: Yes.
27. John: May I ask you something?
28. Mary: Go ahead.
29. John: Can you slide the bolt?
30. Mary: Yes.
31. John: I suggest that you slide the bolt.
32. Mary: All right.

Dialogue C
parallel processing set
John's speed = 1 Mary's speed = 1

1. John: Mary.
2. Mary: Yes.
3. John: I want to suggest a goal.
4. Mary: Go ahead.
5. John: Will you help me get the door open?
6. Mary: By all means.
7. Mary: Shall we make a plan?
8. John: OK.
9. Mary: I suggest that I push the door.
10. John: Mary.
11. Mary: Yes.
12. John: May I ask you something?
13. Mary: Go ahead.
14. John: Is the bolt up?
15. Mary: No.
16. John: Mary.
17. Mary: Yes.
18. John: I want to explain something.
19. Mary: Go ahead.
20. John: If you push the door when the bolt is up, the door changes position.
21. Mary: I see.
22. Mary: I suggest that we get the bolt up and then I push the door.
23. John: All right.
24. John: Shall we make a plan?
25. Mary: OK.
26. John: Mary.
27. Mary: Yes.
28. John: May I ask you something?
29. Mary: Go ahead.
30. John: Can you slide the bolt?
31. Mary: Yes.
32. John: I suggest that you slide the bolt.
33. Mary: All right.

Figure 4.5 Dialogue of the goal of getting the door open with the bolt down, illustrating similar features to fig 4.4 but with three different speed settings. In A, Mary dominates by proposing 3 plans, in B, John dominates by proposing 2 plans, but in C John proposes 1 plan and Mary 2. Discussions about beliefs only occur in A and C.

shows that the number of utterances, the order in which they occur, what gets said and agreed, what is explained, how a plan is realised are all different. Further complications arise when planning gets more involved. Depending on the circumstances,

*** what John's capabilities are ***
1. Goal: Help me get the door open.
2. Position of objects: door: shut: seen, John: out: seen, Mary: undef: undef, bolt: undef: undef,
3. Consequences of events:
   push: If you push the door when the bolt is up, the door changes position.
   slide: If you slide the bolt when you are in, the bolt changes position.
   move: If you move when the door is open, you change position.
4. Range of actions: move: Yes, slide: Yes, push: No,

*** what Mary's capabilities are ***
1. Goal: None
2. Position of objects: door: shut: seen, John: undef: undef, Mary: in: seen, bolt: down: seen,
3. Consequences of events:
   push: If you push the door, the door changes position.
   slide: If you slide the bolt, the bolt changes position.
   move: If you move when the door is open, you change position.
4. Range of actions: move: Yes, slide: Yes, push: Yes,

(State of the world is now [John out, Mary in, bolt down, door shut])

Dialogue A
parallel processing set
John's speed = 3 Mary's speed = 1

1.	John:	Mary.
2.	Mary:	Yes.
3.	John:	I want to suggest a goal.
4.	Mary:	Go ahead.
5.	John:	Will you help me get the door open?
6.	Mary:	By all means.
7.	John:	Shall we make a plan?
8.	Mary:	OK
9.	John:	Mary.
10.	Mary:	Yes.
11.	John:	May I ask you something?
12.	Mary:	Go ahead.
13.	John:	Can you push the door?
14.	Mary:	Yes.
15.	John:	Mary.
16.	Mary:	Yes.
17.	John:	May I ask you something?
18.	Mary:	Go ahead.
19.	John:	Is the bolt up?
20.	Mary:	No.
21.	John:	I suggest that we get the bolt up and then you push the door.
22.	Mary:	All right.
23.	John:	Shall we make a plan?
24.	Mary:	OK
25.	John:	I suggest that we get me in and then I slide the bolt.
26.	Mary:	All right.
27.	John:	Shall we make a plan?
28.	Mary:	John.
29.	John:	Yes.
30.	Mary:	May I ask you something?
31.	John:	Go ahead.
32.	Mary:	Are you in?
33.	John:	No.
34.	John:	Shall we make a plan?
35.	Mary:	OK
36.	John:	I suggest that we get the door open and then I move.
37.	Mary:	All right.
38.	John:	Shall we make a plan?
39.	Mary:	OK
40.	John:	I suggest that we get the bolt up and then you push the door.
41.	Mary:	All right.

Dialogue B
parallel processing set
John's speed = 1 Mary's speed = 3

1.	John:	Mary.
2.	Mary:	Yes.
3.	John:	I want to suggest a goal.
4.	Mary:	Go ahead.
5.	John:	Will you help me get the door open?
6.	Mary:	By all means.
7.	Mary:	Shall we make a plan?
8.	John:	OK
9.	Mary:	I suggest that I push the door.
10.	John:	Mary.
11.	Mary:	Yes.
12.	John:	May I ask you something?
13.	Mary:	Go ahead.
14.	John:	Is the bolt up?
15.	Mary:	No.
16.	John:	Mary.
17.	Mary:	Yes.
18.	John:	I want to explain something.
19.	Mary:	Go ahead.
20.	John:	If you push the door when the bolt is up, the door changes position.
21.	Mary:	I see.
22.	Mary:	I suggest that we get the bolt up and then I push the door.
23.	John:	All right.
24.	Mary:	Shall we make a plan?
25.	John:	OK
26.	Mary:	I suggest that I slide the bolt.
27.	John:	Mary.
28.	Mary:	Yes.
29.	John:	May I ask you something?
30.	Mary:	Go ahead.
31.	John:	Are you in?
32.	Mary:	Yes.
33.	John:	All right.

Figure 4.6 Relative speed settings can prevent the main goal from ever being achieved. Dialogue A contains a circular plan.

agents take the initiative at different points in the dialogue (figure 4.5). Thus a mixture of dominance would arise when the relative speeds of the agents are the same. In some cases the speed settings can determine whether or not the goal is ever achieved, for example a hasty and ignorant agent may direct the dialogue into a circular plan (see figure 4.6).

But whatever the speed settings, an agent must develop an understanding of itself so that it can interrupt its own thinking purposefully. In SUPERPOWER this is achieved with a demon in the executive that ensures that repetitiveness within the joint plan tree does not occur. Thus an important component in addition to circular planning for the understanding of dialogue structure is for agents to be allowed to think, act and speak in parallel, as outlined here. This all important timing in purposeful conversation and action is at the heart of level 4.

4.1.3.1 How do time speeds give rise to changes in dialogue?

From the previous computer dialogue transcripts, it is apparent that a number of features change as a direct consequence of altering the initial relative speed settings of the agents. These include (1) utterance content, (2) dialogue order, (3) utterance count, (4) dialogue outcome, (5) interruption rate and (6) dialogue interaction. I now discuss this in more detail.

(1) The utterance content changes since, in the case of agents with similar knowledge, questions about who is to carry out a plan depends on who is thinking the fastest. Thus "John: Could you push the door ?", when John is fast, would be replaced by "Mary: I suggest I push the door", when Mary is fast.

(2) Dialogue order is affected by the speed settings since agents with different sets of knowledge are liable to want to say things in a different order. An agent who has a great deal of perceptual knowledge but a poor belief system (i.e. causal rules for planning purposes) will want to clarify planning strategies first. An agent with poor perceptual abilities, however, will want to discuss knowledge first. Thus by adjusting the timing accordingly one can determine the order of the dialogue.

(3) A fast and intelligent agent who cannot do many things is going to use additional conversational resources in order to achieve something that it would not otherwise be able to do. For example, a similar agent able to do the task of opening the door would not need to ask the question "Can I push the door?" whereas an agent unable to push the door would have to ask the question "Can you push the door?". Thus the utterance count is affected by the relative speeds of the agents since in the former case one less question needs to be asked.

(4) The dialogue outcome or the final goal states of the agents and the physical state of the world are also affected by the relative speeds of the agents. An interaction between a fast agent who is incapable of achieving the goal and a slow but competent agent may in the end prove unsuccessful if agents are not sufficiently intelligent to recognise when the dialogue is repeating itself. However, on reversing speeds, the dialogue can be successful since the intelligent agent now has control and thus guides it to a predictable outcome. A more intelligent system would detect circularity and act more appropriately.

In some instances the dialogue may fail on one speed setting and succeed on another. For example, see the dialogue in figures 4.9a,4.9b,4.9c, where both robots know nothing about how to do things but can see and do things. One is In, the other is Out, the door is Shut and the bolt is Up. The goal is to get the door Open. In this case, if Mary is fast the task will not succeed if she decides to experiment with the bolt first since she learns correctly about the bolt but fails to appreciate what happens when she pushes the door. If John is fast then sliding the bolt has no effect, since he is Out. Thus pushing the door causes it to open and the main goal is achieved. A more intelligent system would take note of all situations that are true when an action takes place and not just the one that seems the most relevant.

Thus in both these examples, although what I discuss is true for SUPERPOWER as currently implemented, it would not necessarily remain this way if the system were made more advanced since in both cases the dialogue could be made to finish successfully. But my discussion remains the same since I could construct a more complex environment in which the outcome was successful for one speedy agent and not for the other. Thus my conclusions still remain valid even if I were to make the program deal with these two cases in such a way as to make the outcome similar for either speed setting.

(5) Interruption rates can be measured from computer dialogue by considering the number of times the agents switch control from one to another outside the normal adjacency pair sequences of John/Mary/John/Mary etc. Altering the relative speeds of agents introduces artificial control into the dialogue even when the content and situation do not warrant it. Thus, in order for the dialogue to reach a successful outcome one agent will often have to interrupt and supply relevant information. For example, consider a situation where John cannot slide the bolt or push the door and Mary's belief about the world is suspect. When John is speedy he needs to invest resources in asking how to do things, when Mary is speedy John needs to interrupt her since she suggests a plan that will not work. Thus, the mix of

knowledge, skills and timing explain how the interruption rate may change in the model.

(6) This example also illustrates how the dialogue interaction may alter as a direct consequence of the relative speeds. In one situation the dialogue may involve asking your partner about actions while for another situation it could be about discussing beliefs. The only change at the input stage is in the relative speeds of the agents.

4.1.3.2 A Monte Carlo simulation of Power's robot world.

In this section, I provide a more robust demonstration of how the speed settings lead to variations in dialogue. It would not be feasible to test all dialogue variations since there are too many combinations (10^{21}) of inputs possible to the model. We can however summarise the dialogue by choosing input parameters in such a way as to represent the domain as a whole.

In the following simulation, two speed settings were chosen, Fast (referred to as F) and Slow (referred to as S), two belief settings were chosen, one in which the agent knew the effect of the action of pushing the door (referred to as KNW) and the other in which the agent did not know (referred to as UNK). Two perceptual abilities were chosen, one in which the agent can see everything (referred to as PER) and the other in which the agent can only see itself (referred to as IMPER), along with two action abilities, one in which the agent can do everything (referred to as PRA) and the other in which the agent can only move (referred to as IMPRA). John was positioned as In, Mary was positioned as Out, the door was Closed and the bolt was Up. John was given the simple goal of getting the door open[24].

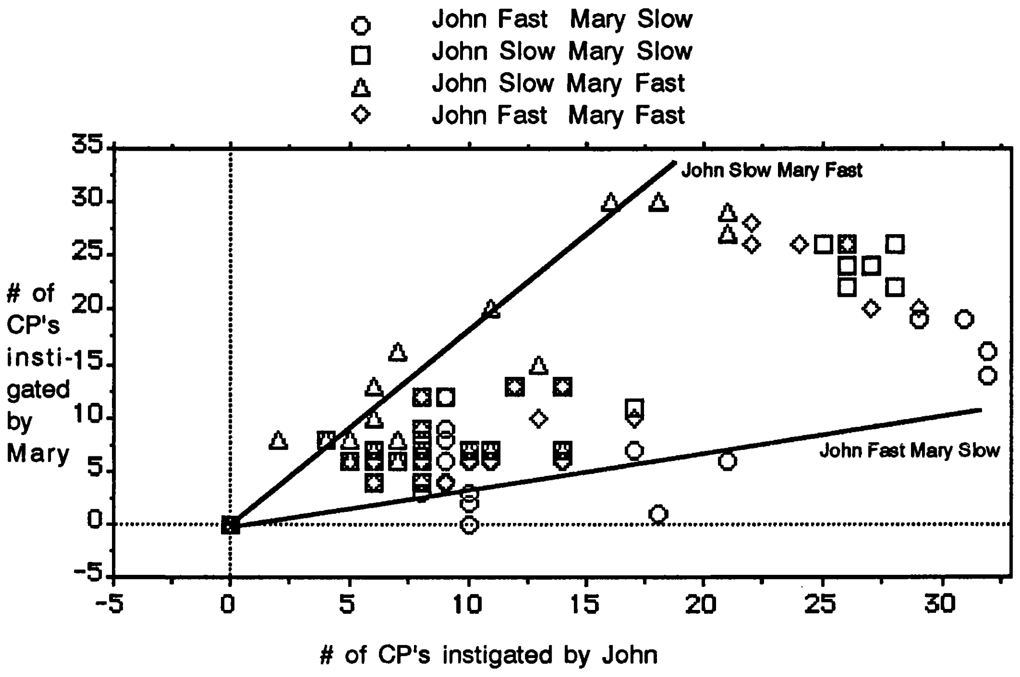
Thus the range of skills related to perception, knowledge and action, with two alternative settings for each skill, generates sixty four groups containing four dialogues for each

---

[24] Total possible dialogue outcomes = 2x2x2x2x5x5x5x5x5x5x4!x4!x4!x4!x4!x4!x4!x4!x3x3x2!x2!x2!x2!x2!x2!x300

Possible Settings	Input parameter	Possible combinations
[bolt up/down door open/shut Mary in/out John in/out]	Positions of objects	2x2x2x2
Push door any nothing undef door bolt up nothing door	Beliefs	5x5x5x5x5x5
[Bolt Door John Mary]	Own Perception	4!x4!
[Bolt Door John Mary]	Perception of partner's perception.	4!x4!
[Push Move Slide]	Own Actions	4!x4!
[Push Move Slide]	Perception of partner's action	4!x4!
[3,2,1]	Relative Speeds	3x3
Door in/out Bolt up/down robot in/out	Goals	2!x2!x2!x2!x2!x2!
push door slide bolt move robot at utterance 1 ....utterance 100	Unexpected Events	~300

of the four different speed settings fast/slow (FS), slow/fast (SF), slow/slow (SS) and fast/fast (FF) for the two agents. FS



Speed Setting	Mean	Standard Deviation	Kurtosis
Total SF	14.1	15.8	.12
Total SS	14.7	16.7	.38
Total FF	14.4	16.2	.33
Total FS	14.8	15.5	.12

Figure 4.7 Scattergram showing distribution of conversational procedures/dialogue by agent and speed setting. This shows (indicated with a regression line) that the combinations of fast/slow slow/fast have a wider distribution than slow/slow fast/fast. Although a one factor ANOVA-Repeated measures (F<1) indicate no significant difference in the overall mean (includes zeros for unsuccessful dialogues) number of CP's there is a clear difference in the kurtosis with more CP's being allocated to the faster agent.

is considered to be a different case from SF because agents are faced with different environmental constraints. Then for the same level of skill the number of conversational procedures were counted, both in total and for the distribution between the agents, and split by the four combinations of speeds. Figure 4.7 shows the overall distribution of conversational procedures instigated by the agents in relation to speed setting. The main result is that more CP's are allocated to the agent with the faster speed setting. Figure 4.8 shows that there is no

Skill level	John Slow/ Mary Fast CP's by John	CP's by Mary	Total	John Fast/ Mary Slow CP's by John	CP's by Mary	Total	John Slow/ Mary Slow CP's by John	CP's by Mary	Total	John Fast/ Mary Fast CP's by John	CP's by Mary	Total
6	2	8	10	10	0	10	6	4	10	6	4	10
5	4	8	12	9	4	13	6	5	11	6	5	11
4	7	10	17	11	5	16	9	7	16	9	7	16
3	11	15	26	17	8	25	14	13	27	13	12	25
2	16	25	41	27	13	40	23	20	43	22	19	41
1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.8 Summary of the number of conversational procedures for a given level of skill, split by the various speed settings and by agent. For example, skill level 3 contains any dialogues that have 3 positive skills and 3 negative skills, a positive skill being one of knowledgeable, practical or perceptive and a negative skill being one of unknowledgeable, impractical or imperceptive. A zero indicates that the dialogue was unsuccessful.

significant (a repeated measures factor analysis of variance revealed that $F<1$) difference in the number of conversational procedures executed per dialogue, for each of the different speed settings and for a given level of overall skill for the agents. There is, however, a significant difference in the distribution of the number of conversational procedures between the agents. More conversational procedures are allocated to the faster agent.

1.	John:	Mary.
2.	Mary:	Yes.
3.	John:	I want to suggest a goal.
4.	Mary:	Go ahead.
5.	John:	Will you help me get the door open?
6.	Mary:	By all means.
7.	Mary:	May I ask you something?
8.	John:	Go ahead.
9.	Mary:	Is the door open?
10.	John:	No.
11.	Mary:	Shall we make a plan?
12.	John:	OK.
13.	Mary:	I can't think of one.
14.	John:	I will then.
15.	John:	I can't think of one.
16.	Mary:	Oh.
17.	Mary:	Shall we do an experiment?
18.	John:	Go ahead.
19.	Mary:	John.
20.	John:	Yes.
21.	Mary:	May I ask you something?
22.	John:	Go ahead.
23.	Mary:	Can you slide the bolt?
24.	John:	Yes.
25.	Mary:	John.
26.	John:	Yes.
27.	Mary:	May I ask you something?
28.	John:	Go ahead.
29.	Mary;	Is the bolt up?
30.	John:	Yes.
31.	Mary:	I suggest that you slide the bolt.
32.	John:	All right.
33.	Mary:	--.
.		.
.		.
.		.
49.	Mary:	--.
		(State of the world is now [John out, Mary in, bolt up, doorshut])
50.	Mary:	--.
51.	John:	I want to tell you something.
52.	Mary:	Go ahead.
53.	John:	I have slid the bolt.
54.	Mary:	I see.
55.	Mary:	May I ask you something?
56.	John:	Go ahead.
57.	Mary:	Is the bolt up?
58.	John:	Yes.
59.	John:	I want to tell you something.
60.	Mary:	Go ahead.
61.	John:	Nothing has happened.
62.	Mary:	I see.
63.	Mary:	Shall we do an experiment?
64.	John:	Go ahead.
65.	Mary:	John.
66.	John:	Yes.
67.	Mary:	May I ask you something?
68.	John:	Go ahead.
69.	Mary:	Can you push the door?
70.	John:	Yes.
71.	Mary:	John.
72.	John:	Yes.
73.	Mary:	May I ask you something?
74.	John:	Go ahead.
75.	Mary:	Is the door open?
76.	John:	No.
77.	Mary:	I suggest that you push the door.
78.	John:	All right.
79.	Mary:	--.
.		.
.		.
.		.
94.	Mary:	--.
		(State of the world is now [door open, John out, Mary in, bolt up])
95.	Mary:	--.
96.	John:	I want to tell you something.
97.	Mary:	Go ahead.
98.	John:	I have pushed the door.
99.	Mary:	I see.
100.	Mary:	May I ask you something?
101.	John:	Go ahead.
102.	Mary:	Is the door open?
103.	John:	Yes.
104.	John:	I want to tell you something.
105.	Mary:	Go ahead.
106.	John:	The door has changed position.
107.	Mary:	I see.
		***** Plan successful *****

Figure 4.9a John's goal of getting the door open. Mary fast John slow. 7 questions are asked.

These results provide a more robust demonstration of what was shown in previous sections. There is, however, one further conclusion from this simulation that I now want to discuss and that is why there is variation within the same skill level. The table in appendix 4 shows the same information split further by skill mix and sorted by level of skill. The results show that, for a given skill level, certain mixes of skills, when combined with a fast/slow speed setting, can shorten or lengthen the dialogue. Figures 4.9a, 4.9b and 4.9c illustrate this

1.	John:	Mary.
2.	Mary:	Yes.
3.	John:	I want to suggest a goal.
4.	Mary:	Go ahead.
5.	John:	Will you help me get the door open?
6.	Mary:	By all means.
7.	John:	Shall we make a plan?
8.	Mary:	John.
9.	John:	Yes.
10.	Mary:	May I ask you something?
11.	John:	Go ahead.
12.	Mary:	Is the door open?
13.	John:	No.
14.	John:	Shall we make a plan?
15.	Mary:	OK.
16.	John:	I can't think of one.
17.	Mary:	I will then.
18.	Mary:	I can't think of one.
19.	John:	Oh.
20.	John:	Shall we do an experiment?
21.	Mary:	Go ahead.
22.	John:	I suggest that I slide the bolt.
23.	Mary:	All right.
		(State of the world is now [John out, Mary in, bolt up, door shut])
24.	John:	I want to tell you something.
25.	Mary:	Go ahead.
26.	John:	I have slid the bolt.
27.	Mary:	I see.
28.	John:	I want to tell you something.
29.	Mary:	Go ahead.
30.	John:	Nothing has happened.
31.	Mary:	John.
32.	John:	Yes.
33.	Mary:	May I ask you something?
34.	John:	Go ahead.
35.	Mary:	Is the bolt up?
36.	John:	Yes.
37.	Mary:	I see.
38.	John:	Shall we do an experiment?
39.	Mary:	Go ahead.
40.	John:	I suggest that I push the door.
41.	Mary:	All right.
		(State of the world is now [door open, John out, Mary in, bolt up])
42.	John:	I want to tell you something.
43.	Mary:	Go ahead.
44.	John:	I have pushed the door.
45.	Mary:	I see.
46.	John:	I want to tell you something.
47.	Mary:	Go ahead.
48.	John:	The door has changed position.
49.	Mary:	John.
50.	John:	Yes.
51.	Mary:	May I ask you something?
52.	John:	Go ahead.
53.	Mary:	Is the door open?
54.	John:	Yes.
55.	Mary:	I see.
		***** Plan successful *****

Figure 4.9b John's goaal of getting the door open. Mary slow John fast , 3 questions are asked.

point. I consider the setting in which both John and Mary are ignorant, John is practical and perceptive, Mary is impractical and imperceptive. Figure 4.9a illustrates the case where John is slow and Mary is fast (5 calls by name, 1 goal, 1 plan, 2 experiments, 7 asks and 4 tells are made). Figure 4.9b

```
 1. John:   Mary.
 2. Mary:   Yes.
 3. John:   I want to suggest a goal.
 4. Mary:   Go ahead.
 5. John:   Will you help me get the door open?
 6. Mary:   By all means.
 7. Mary:   May I ask you something?
 8. John:   Go ahead.
 9. Mary:   Is the door open?
10. John:   No.
11. John:   Shall we make a plan?
12. Mary:   OK.
13. John:   I can't think of one.
14. Mary:   I will then.
15. Mary:   I can't think of one.
16. John:   Oh.
17. John:   Shall we do an experiment?
18. Mary:   Go ahead.
19. John:   I suggest that I slide the bolt.
20. Mary:   All right.
21. Mary:   --.

            (State of the world is now
             [John out, Mary in, bolt up, door shut])

22. John:   I want to tell you something.
23. Mary:   Go ahead.
24. John:   I have slid the bolt.
25. Mary:   I see
26. John:   I want to tell you something.
27. Mary:   Go ahead.
28. John:   Nothing has happened.
29. Mary:   John.
30. John:   Yes.
31. Mary:   May I ask you something?
32. John:   Go ahead.
33. Mary:   Is the bolt up?
34. John:   Yes.
35. Mary:   I see.
36. Mary:   Shall we do an experiment?
37. John:   Go ahead.
38. Mary:   John.
39. John:   Yes.
40. Mary:   May I ask you something?
41. John:   Go ahead.
42. Mary:   Can you push the door?
43. John:   Yes.
44. Mary:   John.
45. John:   Yes.
46. Mary:   May I ask you something?
47. John:   Go ahead.
48. Mary:   Is the door open?
49. John:   No.
50. Mary:   I suggest that you push the door.
51. John:   All right.
52. Mary:   --.

            (State of the world is now
            [door open, John out, Mary in, bolt up])

53. John:   I want to tell you something.
54. Mary:   Go ahead.
55. John:   I have pushed the door.
56. Mary:   I see.
57. John:   I want to tell you something.
58. Mary:   Go ahead
59. John:   The door has changed position.
60. Mary:   John.
61. John:   Yes.
62. Mary:   May I ask you something?
63. John:   Go ahead.
64. Mary:   Is the door open?
65. John:   Yes.
66. Mary:   I see.

            ***** Plan successful *****
```

Figure 4.9c John's goal of getting the door open. Equal speed settings. 5 questions are asked.

illustrates the case where John is fast and Mary is slow (4 calls by name, 1 goal, 1 plan, 2 experiments, 3 asks and 4 tells are made). Figure 4.9c considers the case when the speed settings of the agents are the same (5 calls by name, 1 goal, 1 plan, 2 experiments, 5 asks and 4 tells are made). The only difference between the dialogues is in the number of questions that are asked. This tells us that when skills are kept constant and speed settings are adjusted it is the environment in relation to the agents skill that determines how quickly the dialogue finishes. Similar features were illustrated in the dialogues of figure 4.5 mentioned earlier.

The results from this simulation confirm the findings mentioned in the previous section in a more robust manner. Furthermore the scattergram demonstrates that speeds accentuate the mix of skills and provide new distributions of utterances within dialogue, in terms of length and content, that would otherwise not be seen in AI models of dialogue. Lengthy dialogue occurs when a fast agent has poor skills such as being impractical and ignorant. Brief dialogue occurs when a fast agent has good skills. To this extent the outcome of the simulation produces results that are consistent with our expectations.

4.2 The need for level 4.

In this section I lay down the foundation of all the components that have been considered (in terms of levels) so far in SUPERPOWER and cover level 4 (occupying the airwaves as defined in section 2.2). Issues about how to reason at this level are not covered but ideas about how to proceed are mentioned in the concluding chapter.

In section 4.1, I explained how SUPERPOWER has explored some of the problems arising if agents are to be free to talk, listen or act whenever they want. If an agent must deal with something unexpected then it must reason about it in relation to its own thoughts. So also must an intelligent hearer, if it wishes to speak. In addition, the system reacts to unexpected events instantaneously without thinking by channelling within the executive certain event types such as an action or conversational procedure appropriately.

Thus, I have shown a need for level 4 in SUPERPOWER by illustrating that timing is a critical factor in purposeful dialogue (figures 4.4,4.5,4.6). Next I have provided some groundwork for reasoning intelligently at level 4 by starting to define what is meant by a quantum increment of time. The next stage was to provide a much more explicit theory of how to decide between the demand for external events, including utterances and internal plans. Finally I backed up my findings about how speeds influence dialogue by means of a Monte Carlo simulation. I concluded that speed settings do not affect the dialogue overall, for a given skill level, but that the distribution of the decisions as to which agent instigates the procedures is affected. I also concluded that certain speed settings may accentuate good and bad dialogue for certain skill combinations that relate either favourably or unfavourably with the agents' environment.

In this chapter I have provided a horizontal account of SUPERPOWER with the intention of trying to convey to the reader its maximum capabilities. Now in the penultimate chapter I will consider a vertical interpretation of how the planning mechanisms are invoked at various stages within the dialogue.

Chapter 5.

Kinds of Planning.

5.0 Introduction.

In this chapter, I extend and generalise the planning mechanisms by using all possible combinations of the three components of causal rules (actions, preconditions and effects).

1) Given a rule, an action and a precondition, what can we predict about the corresponding effect? (Normal classical planning)
2) Given a rule, an action and an effect, what can we infer about the corresponding precondition? (Instrumental testing)
3) Given a precondition, an action and an effect, what can we induce about the corresponding rule? (Experimenting)
4) Given a rule, a precondition and an effect, what can we infer about an observed action? (Unexpected or reative planning)

Figure 5.1 The four fundamental planning mechanisms.

Figure 5.1 lists possible ways to reason from planning knowledge. I will show how these generate new requirements for conversational and action-related mechanisms. This, then, provides an exhaustive framework for what conversation is possible in purposeful dialogue. It is the groundwork to level 3.

Although the main concern of this chapter is to explain why these four planning mechanisms form the basis of purposeful dialogue, two additional sections have been included to deal with the problems of conversational stance and the methods for addressing circular planning. Conversational stance dictates what sort of role an agent is going to play. Is it solo, cooperative or somewhere in between? In this system, agents start out in solo and when they realise the goal cannot be achieved they begin to cooperate. New dialogue can also be generated if a role parameter is introduced that increases the level of cooperation every time the main goal fails. The section on circular planning is included to make the simulation mentioned in Ch. 4 feasible. That is to say a demon is needed to detect plans that repeat themselves. These two features are needed to make the system as a whole more robust. They are not, however, the main concern but are mentioned as separate sections later on.

The purpose of this chapter is to discuss the extension of the planning mechanisms that lead to dialogues which justify the distinction between processing a topic[24] (level 2) and processing an item for mutual agreement (level 3). These prerequisites for a satisfactory account of level 3 and to some extent level 2 are expressed as preliminary conclusions in section 5.2.1.

5.1. The four fundamental kinds of joint planning and the need for conversation.

There are only four types of joint planning possible and I now describe how they work in some detail. First I shall

[24] In extended computer dialogue there is a danger that wrong inferences are made because of pragmatic ambiguity (e.g. fig 5.6 summarises the point of the dialogue in which two parts of the tree have the same label [door open] but different plan types (Classical and Instrumental). A reasoning process that wants to detect circularity must identify both the label and the plan type otherwise a valid plan tree would be rejected on the grounds of circularity.

In human dialogue this is dealt with satisfactorily by instrumentally managing topics (the empirical investigation of chapter 2 briefly defines a topic). In SUPERPOWER (ch 4 and 5) it means: A collection of goals that are used to achieve one and only one of the four fundamental types of plans (fig. 5.2). It starts at the beginning of the particular plan type and ends when the top goal is achieved. Thus in some cases it can extend for quite a long time, particularly when more topics are needed to clarify other aspects of the initial topic. In chapter 6 this definition is refined even further to mean a single goal of mutually agreeing something and contains an agenda of items to say or said, to do or done. As this chapter was only a hypothetical design I was unable at this stage to define exactly how this latter definition would avoid pragmatic ambiguity.

describe how they were originally implemented as fixed sequential algorithms within the planner. However in the next chapter I will develop physical and conversational rules, made up of the three components defined in Power (1979), (actions preconditions and effects), that the planner can use for each kind of planning. The four kinds of planning are:

(1) Classical planning as described by Houghton and Power, i.e. selecting actions to achieve an effect.

(2) Instrumental testing in which there is insufficient knowledge. Agents select actions that can reveal the state of an object that cannot, for example, be seen.

(3) Experimental planning in which there is insufficient knowledge about the effects of actions. In this situation the agents try experimenting with actions to discover their effects.

1)	John: I suggest we get the bolt up. Mary: All Right.	Classical Planning
2)	John: Is the bolt up? Mary: I don't know.	Instrumental Testing
3)	John: How do I get the bolt up? Mary: I don't know.	Experimental Planning
4)	John: The door has changed position unexpectedly.	Unexpected Planning

Figure 5.2 The four fundamental planning styles.

(4) Unexpected planning in which agents are inferring unobserved actions from their effects. I only model one event here that happens reasonably infrequently. Planning in a rapidly changing (physical but not conversational) environment has been

addressed, by Wood (1990).

Figure 5.2, illustrates the sort of conversation that might invoke each of these planning mechanisms. The physical planning methods used to process these types of planning are now described in more detail.

5.1.1 Planning phase.

Once the top goal is known, it is passed directly to the planner to find an appropriate course of action. It may be about finding out something, learning the consequences of performing an action, furthering the physical goal directly or doing inferences about something unexpected. Whatever the type, it must fall into one of the four categories described in figure 5.1.

If the task is for a change in the state of the world, then a plan or action may be devised to construct an effect such that if the event were to take place then the original goal would be achieved. This I refer to as classical planning. In this case, the planner finds a rule that produces the desired effect. It checks whether the precondition is true and, if it is true, it then suggests the event related segment of the rule. If the precondition is not true, it finds a rule that satisfies the precondition part, treats this as the main goal and repeats the whole cycle.

If the goal is to find something out, then an action may be found which will enable the system to infer something that is not already known. This is instrumental testing. It follows this sequence: find a rule that contains a precondition that is the same as the unknown information; then suggest the action relevent to this rule. This may be iterative if an agent can not see whether the result of this rule is true.

If the objective is to learn new rules about effects, then by improvising actions whose effects are unknown, the agent's main goal may sometimes be assisted. This is experimental planning. In this case, the planner finds a rule that has an effect that an agent does not know about and returns it.

If something unexpected happens, then it must be assimilated into the main goal. No planning is required before an action occurs, but instead the unexpected event is treated as though an action had just occurred. Then the agents search for a rule whose effect matches the event just observed, and infer that the corresponding action was performed. Pruning then follows.

Thus by extending the "planner" to use the available knowledge about actions in not just one but in all four possible ways, an agent can reason and act to extend its knowledge, as well as achieve physical goals.

5.1.2 Execution.

After an event occurs, assessment is needed in order to establish exactly what has happened.

Plan type	Precondition	event	effect	rule
Classical	Known	Known	Unknown	Known
Testing	Unknown	Known	Known	Known
Experimental	Known	Known	known	Unknown
Unexpected	Known	Unknown	Known	Known

Figure 5.3 The four main kinds of inference achieved by knowing three pieces of information and reasoning correctly about the fourth.

SUPERPOWER proceeds by comparing all information about the state of the world before the action took place with what happened after the action took place. Inferences then take place based on the action, precondition, expected effect, actual effect and rule that were used to construct the plan in the first place. Figure 5.3 summarises the situation.

In classical inferencing, agents are required to assess the consequences of an action. Comparisons are made between the expected and actual effects. If the effect is what was expected, then the goal is considered achieved, and the other agent will be told, if this is appropriate. If this is not yet appropriate, then the rule base is updated accordingly, and the other agent is told when appropriate. If the effect is undefined then the other agent is asked for more information.

In instrumental inferencing agents infer the truth in the world. That is to say, they are required to find out whether or not the situation is true or not and must decide whether the rule used to perform the action produces a change in the state of the world. If it does, then the precondition must be true, otherwise it is false.

In experimental inferencing, agents try to infer how an action affects the main goal. The change in the state of the world reflects the effect of performing such an action in the first place. Thus a rule relating the action to its corresponding effect should be recorded.

In unexpected inferencing, agents analyse events. A change has taken place; that is, an effect has come about which has a rule that explains what action is appropriate to cause this effect. This may or may not affect the main goal. If it does, pruning the plan tree may be necessary and telling the other agent about the action and its effect may also be desirable.

Thus by extending the “inference” process to use the available changes in the state of the world in not just one but four possible ways, an agent can acquire information as well as change the world physically.

5.1.3 An example dialogue

In section 5.1.1 I explained how the system plans actions and that these methods allow the agents a basis for physical mechanisms for achieving a goal. In section 5.1.2 I provided an inference process that allows the system to react appropriately as a consequence of an action. Now in this section I provide a comprehensive example dialogue that illustrates all these mechanisms in operation.[25]

The first five lines are to initialise the program.

```
Compiling /users/Charles/Power/backup/Power.V.5.9/tests.p
Classical/Experimental/Instrumental/Unexpected 1
parallel processing set
John's speed = 3   Mary's speed = 1
(State of the world is now  [John out Mary in bolt up door shut])
```

The structure of the next set of headings is similar to Power; except that in part two, John can see that he is Out ("seen") and does not know, for example, that the bolt is Up ("undef"). He may, however, be "told" that the bolt is Up or he can "infer" it after some action occurs. In SUPERPOWER, agents do not just "know things" or "not know things" as in Houghton and Power's

[25] The reader may compare this transcript with that explained in Chapter 3, listed in appendix 1 in which there are two humans, one blindfolded, and the other trying to negotiate his way through a closed door jammed by a nail.

knowledge representations, but instead, beliefs may have a range of status. The possible states are: “known”/“seen”, “told”, “inferred”, “estimated”, “unknown”, ”undef”.

```
*** what  John's capabilities are ***
1.  Goal: Help me get in.
2.  Position of objects:
    John: out: seen,  Mary: undef: undef,  bolt: undef: undef, door: undef: undef,
3.  Consequences of events:
    slide: undef
    push: If you push the door when the bolt is up, the door changes position.
    move: If you move when the door is open, you change position.
4.  Range of actions:
    move: Yes,  slide: No,  push: No,
```

Mary does not know anything but can do everything and she cannot see John, the bolt or the door. John also cannot see the bolt or the door.

```
*** what  Mary's capabilities are ***
1.  Goal: None
2.  Position of objects:
    Mary: in: seen,  bolt: undef: undef,  door: undef: undef, John: undef: undef,
3.  Consequences of events:
    slide: undef
    push: undef
    move: undef
4.  Range of actions:
    move: Yes,  slide: Yes,  push: Yes,
```

The settings have been chosen to maximise the amount of interaction. Everything proceeds exactly as it did with Power and since John is deliberating three times as fast as Mary, he often takes the initiative in proposing things.

```
1 John: Mary.
2 Mary: Yes.
3 John: I want to suggest a goal.
4 Mary: Go ahead.
```

5 John: Will you help me get in?
6 Mary: By all means.
7 John: Shall we make a plan?
8 Mary: John.
9 John: Yes.
10 Mary: May I ask you something?
11 John: Go ahead.
12 Mary: Are you in?
13 John: No.
14 John: Shall we make a plan?
15 Mary: Ok.
16 John: Mary.
17 Mary: Yes.
18 John: May I ask you something?
19 Mary: Go ahead.
20 John: Is the door open?
21 Mary: I don't know.

At this point, neither agent knows whether or not the door is Open and so they have to digress a little and use their knowledge about action in order to find out whether or not the door is Open.

22 John: Mary.
23 Mary: Yes.
24 John: Shall we do a test?
25 Mary: Go ahead.
26 John: Mary.
27 Mary: Yes.
28 John: May I ask you something?
29 Mary: Go ahead.
30 John: Can you move?
31 Mary: Yes.
32 John: Mary.
33 Mary: Yes.
34 John: May I ask you something?
35 Mary: Go ahead.
36 John: Are you in?
37 Mary: Yes.
38 John: I suggest that you move.
39 Mary: John.
40 John: Yes.
41 Mary: I want to explain something.
42 John: Go ahead.
43 Mary: If you move, nothing happens.
44 John: I disagree.
45 Mary: --.
46 John: If you move when the door is open, you change position.
47 Mary: I see.
48 John: I suggest that you move.
49 Mary: All right.

John now waits for 16 increments of time (blank utterance numbers 50-66) before Mary moves. Meanwhile, and at the same time, Mary is thinking about what she has to do.

50 John:--.
.
.
.
66 John: --.

(State of the world is now [John out Mary in bolt up door shut])

67 Mary: John.
68 John: Yes.
69 Mary: I want to tell you something.
70 John: Go ahead.
71 Mary: I have moved.
72 John: I see.
73 John: Mary.
74 Mary: Yes.
75 John: May I ask you something?
76 Mary: Go ahead.
77 John: Are you in?
78 Mary: Yes.
79 Mary: John.
80 John: Yes.
81 Mary: I want to tell you something.

Mary is about to tell John that the door must be shut when an unexpected event occurs, (the programmer intervenes manually in the simulation), namely the bolt is moved from Up to Down. Analysis reveals nothing...

(State of the world is now [bolt down door shut John out Mary in])

82 John: Mary.
83 Mary: Yes.
84 John: May I ask you something?
85 Mary: Go ahead.
86 John: Are you in?
87 Mary: Yes.

This could be regarded as blatant repetition but is not, since a new event has occurred and the agents need to check that it has not affected the situation. Hence there is a need for better topic management.

88 John: Mary.
89 Mary: Yes.
90 John: I want to tell you something.
91 Mary: Go ahead.
92 John: The door is shut.
93 Mary: I see.

At last, both agents have inferred that the door is Shut and John attempts to finish the plan that he started at line 7. Perhaps line 92 should read "the door must be shut"[26] since that would be a more appropriate utterance for this inference. "The door is shut" would indicate that the utterance was obtained from direct perception. In any case, both agents update their respective knowledge status about the door to "John knows door shut inferred" and "Mary knows door shut told". If an event occurs meanwhile, then this status must go back to "door shut estimated" since it is then no longer absolutely certain that the door is Shut.

94 John: I suggest that we get the door open and then I move.
95 Mary: All right.
96 John: Shall we make a plan?
97 Mary: John.
98 John: Yes.

[26] Palmer (1990) considers this point in terms of dynamic necessity. The basic framework makes the distinction between epistemic and deontic modality and dynamic possibility and necessity. Some aspects of modality can be explained in terms of how the sentences were constructed when in operation. Davey (1978) and Houghton (1986) have dealt with aspects of this but not the particular example I give here that refers to dynamic necessity.

99 Mary: May I ask you something?
100 John: Go ahead.
101 Mary: Is the door open?
102 John: No.

Mary is slow to assess the situation.

103 John: Shall we make a plan?
104 Mary: Ok.
105 John: Mary.
106 Mary: Yes.
107 John: May I ask you something?
108 Mary: Go ahead.
109 John: Can you push the door?
110 Mary: Yes.
111 John: Mary.
112 Mary: Yes.
113 John: May I ask you something?
114 Mary: Go ahead.
115 John: Is the bolt up?
116 Mary: I don't know.

The second instrumental test, in which Mary now pushes the door to see whether or not the bolt is Up or not.

117 John: Mary.
118 Mary: Yes.
119 John: Shall we do a test?
120 Mary: Go ahead.
121 John: I suggest that you push the door.
122 Mary: John.
123 John: Yes.
124 Mary: I want to explain something.
125 John: Go ahead.
126 Mary: If you push the door, nothing happens.
127 John: I disagree.
128 Mary: --.
129 John: If you push the door when the bolt is up, the door changes position.
130 Mary: I see.
131 John: I suggest that you push the door.
132 Mary: All right.
133 John: --.
.
.
.
149 John: --.

(State of the world is now [bolt down door shut John out Mary in])

150 Mary: John.
151 John: Yes.

152 Mary: I want to tell you something.
153 John: Go ahead.
154 Mary: I have pushed the door.
155 John: I see.
156 John: Mary.
157 Mary: Yes.
158 John: May I ask you something?
159 Mary: Go ahead.
160 John: Is the door open?
161 Mary: I don't know.

The problem is further complicated because they cannot assess whether the bolt is Up unless they know whether the door is Open.

162 John: Mary.
163 Mary: Yes.
164 John: Shall we do a test?
165 Mary: Go ahead.
166 John: Mary.
167 Mary: Yes.
168 John: May I ask you something?
169 Mary: Go ahead.
170 John: Are you in?
171 Mary: Yes.
172 John: I suggest that you move.
173 Mary: All right.
174 John: --.
.
.
.
189 John: --.

(State of the world is now [bolt down door shut John out Mary in])

190 Mary: John.
191 John: Yes.
192 Mary: I want to tell you something.
193 John: Go ahead.
194 Mary: I have moved.
195 John: I see.
196 John: Mary.
197 Mary: Yes.
198 John: May I ask you something?
199 Mary: Go ahead.
200 John: Are you in?
201 Mary: Yes.
202 Mary: John.
203 John: Yes.
204 Mary: I want to tell you something.

205 John: Go ahead.
206 Mary: Nothing has happened.
207 John: I see.

Lines 202-207: Mary has established that nothing happened when she moved.

208 John: Mary.
209 Mary: Yes.
210 John: I want to tell you something.
211 Mary: Go ahead.
212 John: The door is shut.
213 Mary: I see.

Lines 208-213: John infers that the door is Shut again because Mary just moved to no avail. The end of a topic occurs when the goal that instigated the new topic has been achieved and deleted from the plan tree. By way of an example given on the next page as a pop11 list, I consider how the topic structure, see also Reichman (1978,1986), might look at line 208. The names of the routines, the main goal and all the sub-goals together with their corresponding actions are indicated in bold. The notation is similar to that used in Power (1979) to describe the control stack. "zrfindout" is a new routine that is used to plan to know things through action and conversation. This illustrates how planning generates the need not only for conversation but also for the management of topics. The point is that John must have at least two topics active in this example since "zrassess" has been called recursively twice: firstly to assess the consequences of Mary pushing in order to know whether or not the bolt is Up and secondly to assess the consequences of Mary moving in order to know whether or not the door is Open. As can be seen from the current control stack.

```
[ [
name zggame expectreply <true>  colour white kind game place 1 entries [2 [] 1 []]]

[  ;;; Assessing the action of Mary moving.
name zrassess kind routine place 11 goaltype test entries [11 [learned] 10 [achieved] 9 [nothing] 8 [done]
7  [Mary in told bolt undef asked door undef asked John out seen] 6 [evt [robot move] sit [door open] res
[robot]] 5 [Mary] 4 [door open] 3 [Mary] 2 [Mary move] 1 [door open] 12 []]]

[  ;;; Finding out if the door is open
name zrfindout kind routine place 2 goaltype test  entries [2 [undef] 1 [door open] 3 []]]

[  ;;; Assessing the action of Mary push the door to see if the bolt is up
name zrassess  kind routine place 9 goaltype test  entries [8 [done] 7 [bolt undef asked door shut inferred
John out seen Mary in told] 6 [evt [robot push] sit [bolt up] res [door]] 5 [door] 4 [bolt up] 3 [Mary] 2 [Mary
push] 1 [bolt up] 12 [] 11 [] 10 [] 9 []]]

[  ;;; Finding out if the bolt is up
name zrfindout kind routine place 2 goaltype test  entries [2 [undef] 1 [bolt up] 3 []]]

[  ;;; Planning to get the door open
name zrplan  kind routine place 4 goaltype plan entries [3 [evt [Mary push] sit [bolt up] res [door]] 2 [evt
[robot push] sit [bolt up] res [door]] 1 [door open] 4 []]]

[  ;;; Asking agreement to get the door open
name zgplan  colour white kind game place 1 expectreply <false> entries [2 [] 1[]]]

[  ;;; Hoping to plan to get the door open
name zrachieve kind routine place 5 goaltype plan entries [4 [sitn] 3 [notyet] 2 [both] 1 [door open] 7 [] 5
[]]]

[  ;;; Main goal of getting John in
name zrbasic kind routine place 8  entries [1 [John in] 8 [] 6 [yes] 4 [failed]] ]
]
```

This captures the point of an utterance, the point of the topic and the point of the dialogue all in one representation.

214 John: Mary.
215 Mary: Yes.
216 John: I want to tell you something.
217 Mary: Go ahead.
218 John: The bolt is down.
219 Mary: I see.

Lines 214-219: Since he knows that the door is Shut and since Mary pushed the door at line 150 John correctly infers that the bolt must be Down. He then returns once again to his original plan that still has some more technical difficulties. This indicates the need to manage topics in order to avoid repetitive dialogue or to interpret an inference in the wrong way.

I have chosen a joint plan tree to represent the main goals and sub-goals (e.g. figure 5.6 later on), a control stack (e.g. diagram after line 214) to represent the point of the dialogue (see Power 1979), and a topic structure containing an agenda of things both said and to be said that defines criteria over a topic and represents the point of an utterance (currently not implemented but as part of the plan tree). This seems to me the most intuitive representation both in terms of how humans conduct conversation and in terms of an implementable model.

220 John: I suggest that we get the bolt up and then you push the door.
221 Mary: All right.
222 John: Shall we make a plan?
223 Mary: Ok.
224 John: I can't think of one.
225 Mary: I will then.
226 Mary: I can't think of one.
227 John: Oh.

Lines 222-227: Neither John nor Mary know how to slide the bolt. They have established at line 94 that they must open the door and then John can move In and after a fair amount of instrumental investigation they know they must get the bolt Up and then Mary should push the door. They don't know what to do next so they improvise; by sliding the bolt, pushing the door and moving randomly to see whether or not any of these actions assist the main plan. Fortunately they choose to slide the bolt first.

228 John: Shall we do an experiment?
229 Mary: Go ahead.
230 John: Mary.
231 Mary: Yes.
232 John: May I ask you something?
233 Mary: Go ahead.
234 John: Can you slide the bolt?
235 Mary: Yes.
236 John: I suggest that you slide the bolt.
237 Mary: All right.
238 John: --.
.
.
.
254 John: --.

(State of the world is now [bolt up door shut John out Mary in])

255 John: --.
256 Mary: I want to tell you something.
257 John: Go ahead.
258 Mary: I have slid the bolt.
259 John: I see.

In order to establish whether or not the bolt is Up, they must repeat some nested instrumental planning. This time, however, progress becomes swifter since the bolt is now Up and therefore pushing and moving helps the main goal as well as establishing knowledge.

260 John: May I ask you something?
261 Mary: Go ahead.
262 John: Is the bolt up?
263 Mary: I don't know.
264 John: Shall we do a test?
265 Mary: Go ahead.
266 John: Mary.
267 Mary: Yes.
268 John: May I ask you something?
269 Mary: Go ahead.

270 John: Is the door open?
271 Mary: I don't know.

Although this question has been raised before at lines 20,101,160 and will be raised later at lines 357 and 426 it is a necessary repetition since knowledge that is "inferred" or "told" is no longer valid after another action. So, after any change in the world, knowledge status must change to "estimated". A clever program would revert this knowledge status to what it was before, if the intervening event was subsequently discovered to be irrelevant.

272 John: Mary.
273 Mary: Yes.
274 John: Shall we do a test?
275 Mary: Go ahead.
276 John: Mary.
277 Mary: Yes.
278 John: May I ask you something?
279 Mary: Go ahead.
280 John: Are you in?
281 Mary: Yes.
282 John: I suggest that you move.
283 Mary: All right.
284 John: --.
.
.
.
300 John: --.

(State of the world is now [bolt up door shut John out Mary in])

301 Mary: John.
302 John: Yes.
303 Mary: I want to tell you something.
304 John: Go ahead.
305 Mary: I have moved.
306 John: I see.
307 John: Mary.
308 Mary: Yes.
309 John: May I ask you something?
310 Mary: Go ahead.

311 John: Are you in?
312 Mary: Yes.
313 Mary: John.
314 John: Yes.
315 Mary: I want to tell you something.
316 John: Go ahead.
317 Mary: Nothing has happened.
318 John: I see.
319 John: Mary.
320 Mary: Yes.
321 John: I want to tell you something.
322 Mary: Go ahead.
323 John: The door is shut.
324 Mary: I see.

Having established that the door is Shut the agents now return to the original problem which was testing whether or not the bolt is Up.

325 John: May I ask you something?
326 Mary: Go ahead.
327 John: Is the bolt up?
328 Mary: I don't know.
329 John: Shall we do a test?
330 Mary: Go ahead.
331 John: I suggest that you push the door.
332 Mary: All right.
333 John: --.
.
.
.
349 John: --.

(State of the world is now [door open John out Mary in bolt up])

350 John: --.
351 Mary: I want to tell you something.
352 John: Go ahead.
353 Mary: I have pushed the door.
354 John: I see.

Now, of course, pushing the door causes the door to move since the bolt is already Up. But they still have not established what happens when you slide the bolt. But it does not matter anyway since some of the higher level goals have been realised. Lines 357 - 368 demonstrate the first example of a convoluted derivative - two goals are achieved in one go.

355 John: May I ask you something?
356 Mary: Go ahead.
357 John: Is the door open?
358 Mary: I don't know.
359 John: Shall we do a test?
360 Mary: Go ahead.
361 John: Mary.
362 Mary: Yes.
363 John: May I ask you something?
364 Mary: Go ahead.
365 John: Are you in?
366 Mary: Yes.
367 John: I suggest that you move.
368 Mary: All right.
369 John: --.
.
.
.
385 John: --.

(State of the world is now [Mary out bolt up door open John out])

386 John: --.
387 Mary: I want to tell you something.
388 John: Go ahead.
389 Mary: I have moved.
390 John: I see.
391 John: May I ask you something?
392 Mary: Go ahead.
393 John: Are you in?
394 Mary: No.
395 Mary: I want to tell you something.
396 John: Go ahead.
397 Mary: I have changed position.
398 John: I see.

Lines 399-421: John is a swift thinker and finally realises that, in the process of planning experimentally to slide the bolt, the main goal can now be accomplished since some of the sub-goals of his original plan were achieved through resolving the positions of the bolt and the door.

399 John: I want to tell you something.
400 Mary: Go ahead.
401 John: The door is open.
402 Mary: I see.
403 Mary: I want to tell you something.
404 John: Go ahead.
405 Mary: The door has changed position.
406 John: I see.
407 John: I want to tell you something.
408 Mary: Go ahead.
409 John: The door is open.
410 Mary: I see.

Now, "the door is open" is repeated at lines 401 and 409 since John has just been told that a change in the state of the world has taken place.

411 John: I want to tell you something.
412 Mary: Go ahead.
413 John: Somebody has pushed the door.
414 Mary: I see.

In this version of SUPERPOWER, no consideration has gone into efficiency. But the problem is more subtle than this. Preconditions for speaking, Cohen and Perrault (1979), are only considered as fixed algorithms within the planner or inferencer.[27]

(State of the world is now [John in Mary out bolt up door open])

415 John: I want to tell you something.
416 Mary: Go ahead.
417 John: I have moved.
418 Mary: I see.
419 John: I want to tell you something.
420 Mary: Go ahead.

[27] It is more likely that this is a bug in SUPERPOWER. It crops up here and in no other place since I probably had not hardwired the correct precondition for speaking at this particular point within the inference process. The problem is more involved than this. The reader may, however, wish to consult the appendix: ROUTINE.ASSESS.p (4 of 5), function zrlesson.

421 John: I have changed position.

Mary has not fully realised the state of the situation and since John has moved she now no longer knows whether or not the door is Open. John however has realised the state of the situation and knows that all he had to do was to move and his main goal would be achieved. Being a co-operative agent, John returns to assisting Mary with her smaller problem of establishing whether or not the door is Open. Thus Mary does not know that John is In and the main goal achieved, since she cannot see him and has other goals still active which take a higher priority over inferring that John is now In.

422 Mary: John.
423 John: Yes.
424 Mary: May I ask you something?
425 John: Go ahead.
426 Mary: Is the door open?
427 John: I don't know.

A change in the state of the world has occurred, so Mary still needs to know if the door is open.

428 John: Mary.
429 Mary: Yes.
430 John: Shall we do a test?
431 Mary: Go ahead.
432 John: Mary.
433 Mary: Yes.
434 John: May I ask you something?
435 Mary: Go ahead.
436 John: Are you in?
437 Mary: No.
438 John: I suggest that you move.
439 Mary: All right.
440 John: --.
.
.
.

```
456 John: --.

(State of the world is now  [Mary in bolt up door open John in])

457 John: --.
458 Mary: I want to tell you something.
459 John: Go ahead.
460 Mary: I have moved.
461 John: I see.
462 John: Mary.
463 Mary: Yes.
464 John: May I ask you something?
465 Mary: Go ahead.
466 John: Are you in?
467 Mary: Yes.
468 John: Mary.
469 Mary: Yes.
470 John: I want to tell you something.
471 Mary: Go ahead.
472 John: You have changed position.
473 Mary: I see.
474 John: Mary.
475 Mary: Yes.
476 John: I want to tell you something.
477 Mary: Go ahead.
478 John: The door is open.
479 Mary: I see.
```

John, having helped Mary with her problem knows that he is In. Mary now realises that the main goal has being achieved and can thus establish that John is now In. The program is stopped automatically when both robots have marked the main goal as achieved.

Thus I have demonstrated an introduction to the kind of stilted but effective conversation that SUPERPOWER achieves so far. I now discuss the two additional features that give the system a further sense of robustness.

5.1.4 Degrees of interaction between agents.

Affecting the conversational stance that agents play in this type of dialogue is the degree of interaction that they are prepared to extend and accept, to and from each other. This is what Shadbolt and Carletta's models refer to as communicative postures. In SUPERPOWER, the stance of the agents can be adjusted to include a number of different proximities of interaction. On one extreme, the agents are uncommitted to one another and perform the main goal on their own. On the other, they are endlessly interactive and require extensive dialogue if ever the goal is to be realised. Thus it would be possible to start the agents off independently and if the goal still can not be accomplished then a role parameter is incremented in such a way as to improve the closeness of interaction. This simple extension has not been fully implemented in version 5.9 discussed here.

In SUPERPOWER there are two main types of stances. Those that focus on the task and those that are heavily engaged in conversation. In the former case agents are largely acting in what Power refers to as solo mode. In the latter case, skills are considerably varied and discussion must ensue. I now discuss these two distinctions in more detail.

5.1.4.1 Action-oriented stance.

Action-oriented stances are those in which there is little need to talk, since the tasks can in the main be performed as though the agents were behaving individually. Only one of the agents is allowed to talk while the other has to proclaim things through action. The top goal may not have been agreed, in which case there is no planning help, but actions and discussions are allowed. Finally, the more normal case that I have considered extensively here allows mutual agreement. This is a simple role parameter that was not fully implemented in version 5.9.

5.1.4.2 Conversation-oriented stance.

In any dialogue, at the outset, the participants must decide upon the level of co-operation that they are prepared to engage in with their partner. With no co-operation[28], Conversational Procedures, Interactional Frames, or items for mutual agreement are not much use. The models described by Shadbolt and Musson (1987) and Clark and Schaefer (1987) probably are useful, however, since the agents make contributions based on their own plans. With perfect co-operation, CP's, IF's or items of mutual agreement are useful since the agents are willing to engage in any conversation knowing that each agent sees things roughly in

[28]Compare this type of interaction with that explained in Chapter 2, listed in appendix 2 about two humans, one of them trying to help the other photocopy an A4 sheet of paper.

the same way. In SUPERPOWER, I have modelled perfect co-operation in that the agents accept all items of mutual agreement. At a later stage, it may be possible to model a mixture of this, since rules about hearing could be incorporated in the same way as rules about planning to converse.

5.1.5 Circular Planning.

Central to the operation of a computer model of dialogue and also to humans being able to conduct conversation purposefully is the general problem of circularity, repetition or cycle-detection. I take repetition to mean utterances that repeat themselves more than once, circularity to mean that the dialogue starts to repeat itself over a period of time and cycle-detection to mean that the computer program enters an infinite loop. This has not been properly addressed in dialogue research. The reason for this is that new sorts of these problems are continually evolving everyday. This section hopes to introduce some of these ideas.

One attribute of conversation is that, however sophisticated the model, it must at some stage be vulnerable to an element of circularity under certain inputs, where the conversation becomes locked into infinite repetitions of the same loop of dialogue.

The following dialogue (Figure 5.4) represents an example

John:	I suggest that we get me In and then I slide the bolt.
Mary:	All right.
John:	I suggest that we get the door open and then I move.
Mary:	All right.
John:	I suggest that we get the bolt up and then I push the door.
Mary:	All right.
John:	I suggest that we get me In and then I slide the bolt.
Mary:	All right.

Figure 5.4 Circular plan.

of a circular plan generated from SUPERPOWER. One solution to this problem is for John to consider whether Mary can perform any parts of the joint plan. In this example, it would have been better if John had asked Mary to slide the bolt. Another solution is for Mary to actively interrupt the plan when she knows she can carry out any of its parts. The more general solution to this problem is for the hearer to interrupt and take control of the joint plan when he or she knows that a circular plan exists. However, the other agent may also then develop the dialogue further into a more complicated circular plan.

In figure 5.4, John's problem was as follows. He wants to slide the bolt but he can not because he is not In. Thus, he needs to move, but is hampered because the door is not Open. So he needs to push the door but would be thwarted because the bolt would not be Up and so it goes on. All plans are plausible to Mary since she never knows, and has no reason to doubt, whether

John's suggestions will work. However, should she be able to sense the same plan being suggested twice she ought to do something and so should John. This shows that there is a need to keep a record not only of the current plan but also of failed plans in a given situation.

Quite often, in joint tasks and in habitual everyday conversation, situations occur in which conversation or action does not seem to be getting anywhere. I am meandering along the street and someone is approaching me. I swerve to the left, and at the same time she shifts in the same direction. We both then retract to the right and so on. Eventually, hopefully, the human agents recognise the multiple repetition of the same actions and take corrective action to break out of this non-progressive loop. There are also examples where these repeating sequences are the only way to achieve the goal. For example, when installing a kitchen sink the pipes have to be judged to be the right distance away from the wall. The basin then slots in on top. The only way to get the pipes in the right position is by trial and error. Superficially this is a circular loop but it differs from figure 4.10 in that the plumber, through his trial and error process, is continuously gathering and acting upon fresh information and refining his actions to achieve the goal. The input he receives varies all the time and increases in subtlety until he gets it right. With the robots in SUPERPOWER the information input can only be a limited choice of options such as “Yes”, “No” or ”I don’t know” and their circular plan is truly an infinite repetition. A more appropriate example might be, for example, the action of going to sleep every day with the goal of not being tired. It is both purposeful and circular but unlike the plumbing example there is no refinement process in the action. The goal can only be achieved through repetition. In joint planning, these are just some of the many situations in everyday life that can create action-oriented circular plans.

Of course the plans just described are problem-oriented with very little conversational content. There are examples which are conversationally-oriented and largely independent of any task. Sometimes they assist, sometimes they do not. For example, saying “good morning” to an eminent lady whenever you happen to meet may offer a long term benefit. On the other hand, the problem arising when you phone a large organisation asking for information about something and they put you through to somebody on extension 1, who then puts you through to extension 3 who then inadvertently refers you back to extension 1, and so on, is a good example of a conversation-oriented circular plan.

Again, the human agent would recognise the repetitive cycle and act to break out of it. In complex everyday life however they do not always manage to do this, for example in cases of drug addiction.

In brief therefore facilities must be provided to enable agents to recognise when repetition is occurring and progress towards the main goal being achieved. When such detection occurs, mechanisms must be in place to both avoid (by detecting when they occur) and evade (by executing alternative action) such habits where and when it is deemed appropriate. However, the problem is still more complex than this since sometimes repetition is useful (e.g. figure 5.5), as in trial and error tasks or

1. John:	Shall we make a plan?
2. Mary:	OK
3. John:	I can't think of one.
4. Mary:	I will then.
5. John:	OK
6. Mary:	I can't think of one.
7. John:	Oh.
8. Mary:	Shall we do an experiment ?

Figure 5.5 A circular plan which can be used constructively to do an experiment.

saying good morning to important people. Sometimes avoiding repetition is not so easy, as in the examples of someone approaching you in a busy street and the phone extensions problem.

A destructive conversation or action-oriented circular plan must be detected and the appropriate goals marked as failed. A constructive solution must be encouraged. The groundwork of this open-ended type of planning is an unproblematic extension (not fully implemented in version 5.9) that could be inserted as a demon in the central executive. This demon serves to trap as

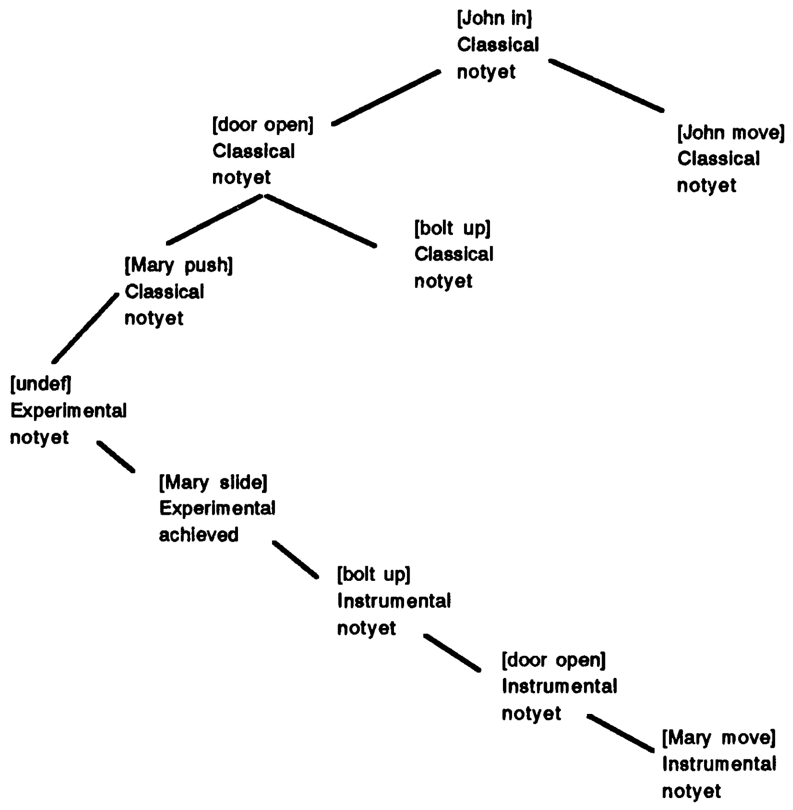


Fig 5.6 Joint plan tree. Line 297, dialogue from section 5.1.3. Illustrating the order in which the physical goal gets executed. The type of planning mechanism and the state of the goal are also needed for unique interpretation.

early as possible any situation in which circularity occurs within the main joint plan tree.

The solution to the problem is by no means simple. Consider, for example, the joint plan tree taken from the dialogue described in section 5.1.3 and illustrated in figure 5.6. Simply detecting when the same sub-goal occurs will not work since the sub-goal may be syntactically similar, e.g. [door open] is mentioned twice, but actually used for a slightly different purpose (in the first case for classical planning and the second for instrumental planning). One improved definition of circularity is a repetitive sequence of plans (i.e. connected for more than one leaf on the tree) counts as circularity. But I have shown that in some circumstances this type of definition can produce useful dialogue. And so the argument goes on.

5.2 Relation to levels.

In this section I lay down the foundation of all the components that have been considered so far in SUPERPOWER in terms of levels. At level 3 I have considerably improved on existing systems in terms of the variety of conversation by considering how a joint planner might utilise all combinations of a simple causal rule about physical action. At level 2 I show how a variety of topics can exist at any one time during the dialogue. At level 1 I make no new contribution. Simple goals such as those described in Houghton and Power are considered. However the need to distinguish between level 2 and level 3 has to be addressed.

5.2.1 Relation to the need for a distinction between level 2 topics and level 3 items of mutual agreement.

In section 5.2, I explored all the four possible types of planning mechanisms that are possible and thus extended SUPERPOWER's conversational variety. In doing so, I illustrated how all four types can generate a need for conversation. Furthermore no planning or inference algorithm can exist without a clear distinction between a topic and a conversational exchange. To avoid doing so would eventually lead to pragmatic ambiguity when comparing the semantics of syntactically similar utterances. It is obvious from the detailed dialogue in section 5.1.3 that given a more complex task the program would either confuse the meaning of an item in one topic with that in another or chunks of dialogue would have to be renegotiated. As currently implemented, a new topic starts when a physical action occurs and ends when the next action occurs. In this implementation, it is only really needed to avoid confusion when inferencing about changes in the state of the world. In the next chapter by defining an item of mutual agreement we can substantially improve on this definition and at the same time, start to reason intelligently about topics as well as items of mutual agreement.

Overall, I have demonstrated the necessity of distinguishing between these three levels and started to provide a framework for thinking about how to implement such a program with explicit consideration of a theory of levels, both in program code and dialogue. I have drawn the reader's attention to how complex human conversation might be generated, by illustrating four basic planning and reasoning tools that provide the ability to reason and act in order to both extend knowledge and also perceive probable states of the world. This, together with simulating in parallel, forms the basis of levels 4 and 3 and indicates the need for level 2. I have also illustrated how a new class of utterances along with dialogue examples might arise.

Chapter 6.

Conclusions.[25]

[25] Parts of this chapter have been adapted from Oatley, Draper and Button (In press), Button and Draper (1990) and Draper and Button (1990). I am indebted to both Steve Draper for providing the idea about conversation is action (Section 6.3) and to Keith Oatley for the discussion about human conversation being about constraint programming (Section 6.2).

6.0 Introduction.

This chapter outlines the achievements of my thesis which can be expressed in terms of the theory of levels mentioned in chapter 2. In that chapter I proposed that there are four distinct conceptual levels to understand. At level 1, the top level, are the goals external to the conversation. At this level SUPERPOWER illustrates these goals chiefly in terms of physical goals but also the following input parameters: relative speeds, beliefs, perception of partner's action, own action skills, perception of partner's perception, own perception, unexpected actions and positions of objects. Next at level 2 are the topics or sub-goals into which the conversation is divided. In chapter 5, I simulated a dialogue in which different sub-goals are active at the same time. At level 3 are the basic exchanges that go into conversation. In chapter 2 I explained how a conversational procedure or an interactional frame such as that described in Power and Houghton is an explanation at this level. At level 4, co-operation in the use of the medium of communication is described in terms of the time slicing covered in depth in chapter 4.

I now summarise the main results and will follow that with a review of the empirical investigation of the thesis and additional observations on program enhancements and further theoretical considerations regarding conversation as action.

6.1 Summary of the main results.

In chapter 2, I provided an empirical investigation that shows that subjects can agree (51%) with the theory that purposeful conversation can be construed as goal-directed with four distinct levels. Level 1 concerns goals external to the conversation, level 2 is about topics that instrumentally manage the dialogue, level 3 is about conversational exchanges and finally level 4 is about occupying the airwaves. One explanation as to why the percentage level of agreement for the analysis of level 3 was not higher is that subjects imagined that the particular exchange which they were trying to code in some way referred to a different goal higher up the plan tree.

In chapter 3 I reviewed early AI work and indicated the shortfalls in terms of these older programs not being able to interact with a copy of themselves. Later in the chapter, I suggested that the more recent goal-directed dialogue systems are restricted in two ways: (a) they can not accommodate two agents thinking independently from one another and (b) the planning algorithms are not always exhaustive for a dialogue system as a whole. For example, none of these systems provides examples of English dialogue that cover the situation where both agents are lacking either a skill or piece of knowledge. I then provided a detailed comparison between two systems (Power and Houghton) that have tried to detail a model of dialogue for a simple domain and a single physical goal. I describe some of their limitations and outlined my proposal to improve on these two systems in terms of time slicing and by incorporating a more exhaustive planning algorithm.

In chapter 4, I described how the relative speeds of the agents affect utterance order, dialogue content, dialogue outcome, number of utterances, distribution of who instigates conversational procedures, what gets said, what gets done, what gets explained and so on. I then went on to look at coordination more generally by generating 256 different dialogues in order to see whether any further pattern emerges. I concluded from the simulation that there is no significant difference in the total number of conversational procedures per dialogue when the speeds of the agents are changed, and the overall skill level is fixed, but that differences emerge in terms of the distribution of conversational procedures instigated by the two agents. Within specific skill levels there is some variation in the total number of conversational procedures executed by the agents when the speeds are changed. Generally a Fast/Slow combination is either better (if environmental factors are favourable) or poorer (if they are not) than the Slow/Slow or Fast/Fast combinations. This led me to conclude that it is the speed skills combined with environmental restrictions and not knowledge, perception or action skills that determine the quality of the interaction. I also concluded that this work must form the basis for the level 4 mentioned in chapter 2.

In chapter 5, I described a generalised planning algorithm that allows agents to have wider conversational skills than reported so far. By considering an exhaustive use of causal rules, I was able to identify the four kinds of planning possible, namely: Classical, Instrumental, Experimental and Unexpected planning. A dialogue containing all these mechanisms was presented to show how a number of different topics can be active at any one time. I then went on to discuss more general issues about circularity and conversational stance and concluded that there is a theoretical difficulty in defining precisely what a circular plan is. I concluded that there is a need to distinguish between levels 2 and 3 in order to avoid pragmatic ambiguity.

In chapters 4 and 5 I outlined the two most important enhancements to the work of Power and Houghton. These concern understanding how the agents can have better knowledge skills and allowing better coordination between agents. It also demonstrates how time slicing is in fact a pragmatic feature of dialogue as a whole and is, moreover what I referred to in chapter 1 as the first principle of coherence which is that there is a definition of an increment of program code that can be reasoned about so as to permit all that can be said. Dialogue also becomes coherent in every way if all four planning algorithms are considered since whatever the initial settings some kind of discussion is generated. The program is, therefore, robust. This relates to the other principle of coherence which is that for every piece of conversation there is a corresponding action method of achieving the same goal.

The remainder of this chapter discusses further conclusions of the empirical work by discussing the limitations of computer models of dialogue and in particular some of the drawbacks of conversational procedures. The next section then discusses possible ways of enhancing SUPERPOWER to cater for problems encountered with the empirical investigation in terms of a generic procedure NEOTELL, which forms the basis of future work that incorporates conversation as action.

6.2 General comments relating to the empirical investigation.

The fact that I was not able to achieve better than 51%

agreement in the empirical investigation in chapter 2 indicates that naturally-occurring human conversation might be more complex than anything generated computationally. At its simplest one of Power's agents will generate utterances designed to agree about a goal, a plan or a belief but will not generate compounds. As the derivations from Power's program function at present each conversational procedure has a single objective which exits when this objective is achieved. Matters that are irrelevant to building the plan tree and executing the plan in a strictly piecemeal fashion simply do not occur. Let me illustrate this with a piece of dialogue from SUPERPOWER

```
16  JOHN:   [Mary.
17   MARY: Yes.]
18 JOHN:   [May I ask you something.
19  MARY: Go ahead.]
20  JOHN:   [Is the door open.
21  MARY: I  don't  know.]
22 JOHN:     [Mary.
23 MARY:    Yes.]
24  JOHN:   [Shall we do a test.]
25 MARY:   Go  ahead.]
26 JOHN :    [I suggest I move]
27 MARY :    [okay]
```

Figure 6.1 Sample code for an instrumental test.

enabling agents to carry out test actions on the world in order to observe effects, and hence to improve their beliefs. The segment, shown in figure 6.1, is from near the beginning of a run of the program, and starts after a goal of getting John In has been agreed.

Both agents are blind, and it has been established by Mary that the goal is not already achieved. Utterances 16 and 17 are a housekeeping exchange — John calls Mary's attention, interrupting any private thoughts she might have. She responds, and this indicates that she will co-operate in the next conversational procedure. Utterance 18 is a request to co-operate on a lower level goal. Mary assents, so that the exchange delivers an agreement to co-operate. Then 20 and 21 constitute a further housekeeping exchange. Utterance 22 is John's opening of an exchange about a plan to test the world by doing an action and finding out what happens... which they go on to do next.

Notice that in the computational example each conversational procedure is designed to achieve exactly one increment of agreement, exclusively either a piece of housekeeping, or something about a goal, a plan or a belief. Moreover, the utterances are driven by the building of the plan tree, with each piece of the agenda for the construction of a plan tree being negotiated explicitly. So if a goal is agreed, then the question arises 'Is it achieved yet?' If a precondition is necessary for a plan, then the agents will see if they know that it is satisfied, if not, as in this sequence they each ask the other, and if the state is still not known, they start to try and test actions. In this case the agents plan for John to move, and if he changes position from Out to In, they will know the door is Open. Few inferences are made, and therefore no misunderstandings occur. Moreover a reader of the transcript, trained in the categorisation system I have suggested in figure 2.1, would have no difficulty in categorising each exchange.

Compare this with the human conversation about photocopying (see appendix 2). First John asks about a goal. All seems straightforward except that 'photocopying' is a complex activity. Unlike utterances about goals in the agents' conversation, 'do the photocopying' does not specify any exact state of the world. It may be for this reason that Mary asks for more details about John's goal: 'Double-sided?' she asks. John seems not to understand fully. His response, 'Yes, and then the other side' seems at least to introduce unnecessary redundancy. Next, in utterance 4, Mary makes a characteristically human remark: 'I don't know, some or...' It is difficult to interpret this. It would indeed be impossible except for the context. In fact she is politely trying to stop John from being too precipitate, and hence wasting a copy, while at the same time she tries to understand what it is that John really wants. She understands that in asking for her to agree on a goal, he does not know everything he might need to know for any specific goal to be described accurately. For instance, is he asking for a general course on photocopying, or a brief tutorial about some aspect, or just for help with a single item? And what does he know about photocopying already?

John, it turns out, though having apparently negotiated a goal, also has another, which he has not specified. He knows about single-sided photocopying, and thinks that double-sided copying is a simple extension of it, perhaps just involving an extra button press. He wants to photocopy just a single page, double-sided. His other goals are (a) that he wants to transfer his previous skills, unmentioned in the conversation, in a direct way and (b) that he wants to do the copy quickly and press the buttons immediately. So instead of specifying all his goals and asking about Mary's at the beginning of the dialogue, as a computer program might, he proceeds, firstly assuming that Mary knows that he knows how to do single-sided photocopying, secondly as though Mary knows that he is only learning, thirdly as though Mary would agree with him that the problem is a simple matter involving a small amount of practical guidance, as is the case in most “normal business environments”.

Despite these misunderstandings, which emerge gradually in the early part of the conversation, the conversation is in the end successful, and a version of John's goal is achieved. Mary succeeds in another goal, namely giving John part of a general course on photocopying which she believes was implied by his initial request. Thus in the end three goals are achieved: (a) getting John's double sided original copied onto a double-sided copy, (b) John being taught that topological considerations must be taken into account with double-sided copying in order, for example, not to get the copy on one side upside down in relation to the other, and (c) John being shown where various things happen, for example that in the course of this kind of copying, the page with a copy on one side is collected in a special hopper, and then runs through the machine again but upside down to receive the copy of the second side. What happens, then, is that John and Mary sometimes accomplish pieces of agreement about more than one goal in the same exchange, and they sometimes accomplish pieces of agreement about plans and beliefs in the same exchange.

So, for example, Mary's utterance 10 is about a belief, but also has a purposive element, first so that she may teach John about topological issues and also possibly to proceed with the immediate plan. So when it comes to coding such an exchange it might be interpreted as a belief or a suggestion to do something.

It is the ambiguity of interpretation of many, perhaps most, utterances in human dialogue that makes it difficult to assign human exchanges to a unique category. This ambiguity is reflected in the several different codes assigned by the eighteen subjects to each exchange. These categorisations were the outcome of extensive discussions about each exchange in the two transcripts presented here. Perhaps the reader will have found some trouble in agreeing with all the categorisations suggested. The necessity for discussion, and the possibility of disagreement with the judgments by readers can therefore be taken as further evidence for the assertion that human utterances, unlike those of Power's program, while they increase agreement in general, are ambiguous, and can not always be uniquely assigned to categories of agreement.

Similarly in the photocopying transcript, in discussing the consequences of a plan, Mary's utterance 20 and John's reply indicate Mary's knowledge of how the machine works. Apparently the minimal increment of the immediate plan has been agreed. Other unagreed goals, however, for example of training John about the topological issues, are still alive.

In judging the transcript about photocopying, the subjects used the housekeeping category 13% of the time. Here again, although I coded none of the exchanges in this way, there are interpretations of some of them which have the quality of meta-comments on the conversation, and it is understandable that some subjects thought this was the salient aspect in some exchanges. So Mary's interrupted utterance 4, 'I don't know, some or...'; also exchanges 14, 15, 16 starting with Mary's 'Pardon...'; and also the single word exchange 27 by John, 'Right', were all judged by many or most of subjects to be Housekeeping. I decided to minimise the number of exchanges that I assigned to this category, and had judged all these as serving to achieve increments of agreement about plans.

The data shows, in general, that whereas all but one subject was clear that the first exchange was about a goal, subjects did not achieve more than 72% agreement with us on any other exchange. Subjects are not, in other words, unanimous about whether to code exchanges as beliefs, plans or compound beliefs plus plans. Normally in psychological studies which require content analysis, lack of agreement is a signal to define categories more rigorously, or improve the subjects' training. At first I did this: category definitions went through a long series of improvements, and I tried several training methods, before settling on the rather elaborate training scheme described here.

I have concluded, however, that it is not the procedures that are at fault. Subjects are correct in their lack of unanimity about their judgments. Their variety of judgments reflects the variety of interpretations of human utterances. Making a decision about whether or not there is any element of purpose in an exchange depends on what interpretation one makes of the utterance. This was not a failure of training, or of conceptualisation. Rather, even in purposeful dialogues, many human utterances are multiply ambiguous, and they carry variable amounts of weight about beliefs and plans, and sometimes other components. If one is making forced choices about categories, the choice depends crucially on which of several interpretations seems salient. By contrast since each utterance of Power's program is designed to achieve an agreement exclusively about housekeeping, or a goal, or a plan, or a belief, the category of the utterance is unambiguous.

In summary, then, Power's work does indeed offer a basis for the structure of purposeful conversation. His proposal offers a viable and basically convincing theory of how utterances are connected together in conversation. There is no better theory about the connectedness of conversation. Furthermore I have shown that people are moderately successful in being able to assign categories to the atomic units of conversation, at the third level of the goal hierarchy that I have postulated to clarify the basis of this connectedness.

This study also reveals, however, several important ways in which human conversants differ from Power's agents. First, whereas computational goals must always be stated exactly, human goals may be stated very vaguely. Secondly, whereas Power's program and its derivatives are driven by just one top level goal that is agreed and then planned for, human conversants typically have several active goals at any time. This seems to make it more likely that each utterance has not just one interpretation, but several. The hearer of the utterance may choose an interpretation, and allow just one increment to be agreed, but equally the other goals held by either partner may remain active and able to influence expressions and interpretations later in the conversation. Conversation is capable of carrying forward these multiple meanings, to be clarified and elaborated in the light of further utterances by both participants. To understand conversation, then, one must be alert to a variety of interpretations of each utterance. Thus categorisation of exchanges is not straightforward. In fact categorising any one utterance, even in the context of others is rather a hard task, since a judge is constantly in conflict about which interpretation to choose. The variety of interpretations by these eighteen subjects indicates the range over which each exchange can be understood.

Though the variety of interpretations of human utterances has been discussed widely by ethnomethodologists, for example Garfinkel (1967), I hope here to have brought this issue into the context of Cognitive Science so that the structure of conversation is recognisable simultaneously from both a computational and a human perspective.

The developments reported here derive from a tradition of Artificial Intelligence research on planning, in which the usual assumption is that goals are pursued one at a time. The assumption fits broadly with the mainstream of Computer Science and with typical planning algorithms in Artificial Intelligence which search exhaustively for one solution at a time. Computational procedures are usually functions that begin from a certain initial condition, as specified by their arguments, or in planning, from the state of the world plus a desired goal state. They then compute a definite end point such as a result, or in planning, a plan to achieve the goal. The conclusion here is that even though purposeful conversation can fruitfully be construed as a kind of planning, and seen as constructed from units which are conversational procedures, it seems that the situation is not so simple.

First, people may have more than one goal at any given level. They then seek plans that simultaneously achieve several or all of them. In conversation, it is possible to cover a number of related points in a single utterance. It is even usual to achieve a range of effects, for instance to inform, warn, and show concern, in one utterance or short stretch of conversation. Artificial intelligence work exploring this problem of pursuing multiple goals is not wholly absent — see, for instance Wilensky, (1983), or Elsom-Cook, (1984), but these are not yet standard. More specifically in dialogue, balancing resources between listening and speaking is an ongoing human problem that current computer models of dialogue have not yet addressed. That is to say, no account has been offered as to how it is possible to plan a speech act not only in relation to planning one's own goals (as do those proposed by Austin, Searle, Power, Houghton, Cohen and Perrault) but also with respect to the goals of the hearer. More generally, the hearer always has two choices, either to respond directly or to respond with respect to private plans. In either case it is more a question of how many resources one wants to put in to listening which is also determined by the constraints of the situation.

Secondly, unlike the idealised situations that much Artificial Intelligence planning addresses, in which the effects of actions are fully known in advance, in conversation speakers typically do not know enough to construct reliable plans, and have to elaborate creatively from the situation as it has developed so far (Suchman, 1987). When we converse the main source of uncertainty is our partner. We do not know how that person will respond to an utterance. They may decline our choice of topic, not know the answer to a question, or they may spring a new piece of information on us that changes the goal we thought we had. This lack of knowledge is a major shortcoming of the theory of speech acts, in which it is assumed that effects of a speech act can be reliably known. It is also a problem with Power's approach.

Attempts to overcome this problem have been made by Appelt (1987) who makes the interesting point that we can at least make an assessment as to the success criteria of an effect based on the referents of the goal. Thus, when we speak although we cannot predict entirely the effect of what we say we do presumably make some predictions about the likely alternatives an utterance might have by the goals it is intended to serve. For example if I want to say “Is the bolt Up?” I know that the likely response is one of “Yes”, “No” or “I don’t know”. Thus, as Appelt clearly demonstrates, although we do not know for sure the actual effect of a speech act we have a good idea about the likely alternative responses to it. The careful speaker would not otherwise have made the utterance in the first place.

By contrast, the human conversations I have recorded show people constructing a joint plan by adding constraints rather than, as originally postulated, by the successive agreement or rejection of small definite proposals. Utterances are ambiguous not only to third-party listeners, but to the conversants themselves. Utterances are ambiguous as to what is meant, as well as to which goal or goals they are related to. Succeeding utterances often serve less to accept or reject proposals, than to allow the conversants to discover together what they themselves mean. A plan emerges from the process of clarification. Thus rather than traditional programming based on functions, conversation is like constraint-programming in which constraints are added and it is left to the system to resolve them and find a solution consistent with them all.

This conclusion, however, is made with reservation since both the transcripts used perhaps led me to it and in both cases the participants did not fully understand the main goal before they started the dialogue. Further transcripts would be needed to confirm this interpretation in cases where the main goal is fully understood. In this situation the theory would be more likely to be correct.

Nevertheless, if this interpretation is upheld, then we can still understand conversation in terms of levels of goals, but some of the goals will not form a strict hierarchy. Furthermore, although the conversational procedure is a conception that is superior to those of other atomic units of conversation, such as speech acts or adjacency pairs, I have revised my view of the effects that a conversational procedure has. It is clear that such procedures need not exit with a small increment of definite agreement, which is then fully established. Instead, an exchange accomplishes a constraint on how the conversation is to be understood. Thus in the photocopying transcript, Mary's utterance 12: 'Yer, it must be in sequence' constrains John's understanding of why Mary seems not to have been as straightforward in her instructions as he expected, and in the second part of this utterance: 'I suppose if you...', she indicates explicitly that she is continuing to search for interpretations of what has been going on, in order to make sense of the interaction so far. John's utterance 13 'Right, well I'm only doing one sheet' further constrains for Mary her understanding of why John has been acting as he has.

Although constraint satisfaction has been taken up by a number of AI workers who are trying to simulate dialogue it could hardly be considered to be a complete theory of dialogue since it can not explain how it is that humans jump from one level of conversation to another in dialogue. Even if it was the theory, attempting a computational account of how to speak in relation to what has just been said may be about doing away with increments of agreement or conversational procedures altogether, and replacing them with a process of interpreting each utterance by the constraints of the overall joint plan, rather than just responding to the initial goal of the exchange as current AI programs do.

Some formalisms in planning have been specifically designed to deal with this very problem. For example, in single-agent planning, Lansky (1988) shows how sensible plans can be devised by choosing from existing plans according to what she refers to as constraint satisfaction. Another way to deal with this is to think of dialogue as the efficient exchange of intentions that are chosen using game theory (see Shadbolt 1984).

In the next section I will show that constraint programming can also be dealt with by considering a theory that relates conversation and action more closely than previously discussed.

6.3 General comments relating to program enhancements.

Suchman (1987) claims that models run into difficulties when conversation is required to be developed *in situ*. Coates (1990) explains that models cannot explain how it is that utterances can be developed by more than one speaker and that the underlying intention is only known to participants after the utterance has taken place and not before (see also Searle 1990). My empirical work on constraints discussed in the previous section also supports these two views of the apparent impossibility of a model that provides a satisfactory explanation as to how it can react spontaneously to these types of conditions.

I now propose a system design that will address these problems.

6.3.1 Conversation as action.

Austin (1962, 1970) established the idea that utterances should be analysed as actions. For those working in AI, this constitutes an implicit but long-standing challenge. AI work in planning seeks to develop a particular and detailed model of what it is to act rationally and purposefully, that is to choose and execute actions that are calculated to achieve the agent's goals. Can a theory of utterances be developed that develops Austin's basic idea in this way? Such a theory will be a theory of (a kind of) conversation, both because utterances have an effect only through the presence and cooperation of a hearer or co-conversationalist, and because a planning theory is essentially a theory of how basic actions (in this case utterances) are strung together in a sequence to achieve some overall effect not achievable by a single action. We must therefore be concerned with a theory of conversation, not of isolated speakers or isolated utterances: the latter kind of theory would be a theory of certain actions, but not of purposeful, planned action sequences.

A theory of purposeful conversation should aim to develop Austin's view by showing how utterances can be considered as actions in exactly the same way as physical actions. The agent must be able to achieve goals by acting, and to choose actions by generating a plan linking a set of actions to a final goal. Thus in a planning theory of conversation the agent must be able to achieve goals by conversing, and to generate a plan integrating conversational with physical actions and effects. A theory will most fully achieve these aims if a single planner reasons about both physical and conversational actions, and if for every goal and sub-goal there is both a conversational and a non-conversational method of achieving it. For instance the goal of moving a block might be achieved either by pushing it oneself, or by persuading another agent to move it on one's behalf. The goal of getting information might be achieved either by asking a question or by examining the world through perception and experiment. Similarly the goal of communicating information can be done either verbally by answering a question, or at times by pointing to the answer or demonstrating it. A call by name would have the equivalent action of a tap on the shoulder. Even mutually agreeing an item can often be achieved non-verbally - for instance by catching someone's eye as you approach a door while heavily laden. However this kind of example would then depend heavily on the respondant making correct goal inferences. This important aspect on conversation has been neglected here since the focus is intended to be on explicitness rather than efficiency. Thus, whatever the conversational feature there is an equivalent alternative by way of action.

Even in stating these aims, some of what must be involved begins to emerge. For instance the set of possible states and goals that must be explicitly modelled includes states of knowledge or belief. The main effect of many conversational actions is to change what an agent knows. Furthermore in an integrated theory such states need to play the role not only of effects but also of goals. The planner must be capable of recognising the need for knowledge as a prerequisite for achieving other goals so as to generate plans that include acquiring it purposefully. Since knowledge is above all a prerequisite for planning, the theory must include meta-planning (planning how to plan). This also entails that the theory must deal with mixed planning and execution (in contrast to early simple AI planning, where all the planning was done in a first phase, followed by all the execution), since actions that acquire information must be executed before further planning can proceed. Mixed planning and execution is also required to deal with surprises, that is events that the agent did not cause or predict. Conversation intrinsically involves surprises because hearers cannot always know when the other will speak or what kind of thing they will say. Having knowledge as a goal that can be planned for, further suggests that perception should be treated as an explicit action. Finally, conversation is a coordinated activity and the theory must deal with the nature of the coordination. Is it implicit or is it planned and negotiated like the content of mutually agreed physical plans? If it is wholly implicit, then the conversation cannot be said to be planned in any deep sense. Contrariwise, however an infinite regress threatens any attempt to make the conversation wholly explicit e.g. asking agreement to having a conversation to agree a plan .... In what sense are hearers engaging in planned action if they acquiesce in responding to an unexpected and unsolicited utterance?

6.3.2 Four levels of goals.

In approaching a solution to these aims and problems, it seems crucial to use the distinction mentioned in chapter 2 between the different levels of goal which the agents may have. A complete planning theory would link actions to goals at all levels to the top level, level 1, the external goal which an agent brings to the conversation (hoping to achieve it). For instance, a robot might begin with the goal of getting into another room and undertake a conversation in order to enlist help with this. (In more complex models, not developed here, agents might have multiple such goals and attempt to satisfy more than one at a time; or might have traits such as a homing instinct that tend to generate specific goals in a given situation.)

At the centre of a theory of conversation is the level of what can be achieved in an elementary unit of conversation (level 3). An obvious candidate for these elementary actions is the speech act. However this does not capture the essentially two-agent nature of the unit. Something closer to an adjacency pair seems appropriate. I will define a level 3 goal to be the achievement in a single utterance pair (half of which may sometimes be silent) of mutual agreement of a single item: a joint goal, a planned action, a (belief about a) fact. I will also define a level 2 goal in the same way (as the mutual agreement of a single item) except that it may take a prolonged interchange. This might be because the particular agents do not have a conversational method for achieving it directly, or more probably because initial attempts fail, that is a plan may require justification before being accepted by the other. Thus level 2 goals organise stretches of conversation often consisting of a number of level 3 actions, some of which fail (by not achieving mutual agreement of their content). This organisation corresponds to some of the intuitive notion of a topic (a number of utterances concerning the same thing), but differs from the more declarative treatments of that notion which have been developed to account for such things as anaphoric reference.

Finally level 4 will refer to those aspects of turn taking and coordination below level 3. For instance work on pauses (e.g. Bennett 1981) belongs here. From a planning perspective, these goals concern the management of the medium. One aspect of this is agreeing vocabulary and methods of referring. Another is turn-taking in the sense of not physically interrupting each other. If literal interruption occurs, agents will not be able to hear the words the other utters. This is distinct from the turn-taking problem at higher levels. One such problem concerns how to tell if an utterance is the reply to an earlier one, and if so which one. This is the level 3 problem of sorting utterances into "adjacency pairs" even when they are not adjacent. Another is the level 2 issue of who gets to determine the topic, and the management of topic-nesting and switching. The level 4 issue is strictly about physical interruption, and is bound to the medium, for example there is no such issue when letters are exchanged since the fact of letters crossing in the mail does not render them illegible, nor is there such an issue in current computer interfaces that allow the user to type input simultaneously with system output scrolling on the screen.

In this model there will be only a single generic action at level 3[26] . This is an exchange or conversational procedure called NEOTELL. All other conversational actions are constructed out of it. In NEOTELL, the speaker proposes (tells) an elementary item of some type, and the other agent accepts or rejects it. In the former case, the item becomes mutually agreed. In the latter, the action fails and there is no direct effect. Thus asserting a fact, proposing a plan, and suggesting a joint goal are all cases of this generalised "tell" with different types of

[26] Unlike Power who had 7 CP's and Houghton who had 4 IF's.

item. Questions are treated as two consecutive NEOTELL's — as a request to tell a fact: White proposes as a joint conversational goal that Black tells a fact, Black agrees, Black tells the fact, White agrees (or not, if Black's factual assertion is somehow inconsistent with White's existing beliefs). In some instances, neither participant knows how to accept or reject an item. For example, neither robot may be able to see whether the door is Open. In which case, the task is then for the robots to find out whether the door is Open. Alternatively they may not know what the bolt is for in which case the task is to perform an experiment in order to discover what physical effect the bolt may have. Perhaps the most comparable literature at this level, in Psycholinguistics, would be Sacks, Schegloff and Jefferson (1974) or Clark and Schaefer (1987).

An agent's planner will generate conversational goals of the basic type of achieving mutual agreement about an item. For instance to agree a plan to open the door. The planner then searches for a way of achieving this. It may find a single action to do this and propose the action via NEOTELL. If this succeeds, a single level 3 action was enough. However if it fails (for example the other agent objects to the plan proposal), the goal persists as a level 2 goal until the agents find a way of satisfying it. For instance the other agent may propose an alternative plan if it is accepted as mutual the goal of agreeing a plan. The first agent may ask the other to propose a plan, that is, the first direct conversational plan of proposing a physical plan failed so the second plan is analogous to a question and consists of a request to propose a plan, followed by the proposal. Longer sequences may ensue if other physical plans can be found to propose.

Both planning and inferring at all stages will be done whether an item is a physical or conversational action. Thus the task of the planner, if the goal is conversational, is to plan how to speak by mutually agreeing an item. If on the other hand it is to shut the door then the task of the planner is to perform this action. Likewise, the inference task, if the current action is physical, is to assess the changes in the state of the world. If on the other hand the current action was a joint conversational action to discuss whether the door is Open, then the inference task would be to see whether any goals on the joint plan tree had been achieved. It should also compare the new constraint with the old knowledge states of the world and infer new facts about the world as a direct consequence of the joint conversational action that has just taken place.

Thus, the program should have a number of components: a planner that plans physical and conversational actions, an inference module that assesses the effects of physical and conversational actions, a joint plan tree that represents the point of an utterance or physical plan, an agenda of items[27] already said or to say or do or to be done which together represent the state of the topic and a control stack that represents the state of the dialogue and the status of the overall goal.

Knowledge representation should be provided for conversational rules about how to mutually agree something together with physical rules about how to do something, how to cause an effect, what to do if something unexpected happens and how to find something out. Meta-rules are also provided in order to make decisions about when to plan, when to infer when to act and when to talk. For example if an inference is made about a change in the state of the world and it is known that the other robot does not know about it, it must start talking rather than continue inferring or planning.

Thus level 1 should be thought of in terms of the goals that are brought to the conversation, the planning algorithm and knowledge representation that acts upon the goals and sub-goals that participants must achieve. Level 3's only concern is to mutually agree items. Level 2 is automatically generated by

[27] This is what Searle (1983,1990), Cohen and Levesque (1990) refer to as intentionality. For a criticism of similar definitions see Bratman (1987).

the mechanism described: any sub-goal will persist until achieved, even if this requires a sequence of level 3 actions (NEOTELL's). Such a sequence corresponds to a topic so the sub-goal corresponds to a level 2 goal.

The program should distinguish the control stack whose purpose is to control the whole dialogue from an agenda whose purpose is to coordinate the execution of a topic and from a joint plan tree whose purpose is simply to decide what to do next (which can be speaking, acting, planning or inferring). A topic begins when a sub-goal appears on the plan tree and a sub-goal is defined as any situation that needs to be achieved apart from the main goal. It may be finding something out or having a situation to be true. A topic ends when the sub-goal is achieved.

In Linguistics, unlike previous work that either considered how utterances are generated in isolation, see Searle (1969, 1975), Cohen and Perrault (1979) or in which a limited number of dialogue transcripts were analysed for similar patterns, see Clark and Schaefer (1987), Sacks, Schegloff and Jefferson (1974), I argue that all utterances can be generated by considering conversation as action through dialogue. This fundamental principle can then be applied in an Artificial Intelligence simulation, by considering conversational structure in terms of levels.

John's speed = 3   Mary's speed = 1

*** what John's capabilities are ***

1. Goal: None

2. Position of objects:
Mary: In: seen, bolt: undef: undef, door: undef: undef, zggoal: []: agreed, zgplan: []: agreed, zgexperiment: []: agreed, zgtest: []: agreed, zgask: []: agreed, zgtell: []: agreed, zggame: []: agreed, zgrule: []: agreed, John: out: seen,

3. Consequences of events:
If you [[perform [robot slide]]] when [[is[any]] then [[changes[Undef]]]
If you [[perform [robot push)] when [[is[any]] then [[changes[nothing]]]
If you[[perform [robot move)] when [[is[door is open]]] then [[changes[robot]]
If you [[perform [robot slide]]] when [[cansee [bolt]]] then [[know [robot]]]
If you [[perform [robot push]]] when [[cansee [door]]] then [[know [bolt]]]
If you [[perform [robot move]]] when [[cansee [robot]]] then [[know [door]]]
If you [[perform [robot slide]]] when [[cando [bolt]]] then [[knowhow [bolt]]]
If you [[perform [robot push]]] when [[cando [door]]] then [[knowhow [door]]]
If you [[perform [robot move]]] when [[cando [robot]]] then [[knowhow [robot]]]
If you [[Neotell [Y]]] when [[Gricean [robot]]] then [[MutAgree [X]]]

4. Range of actions: move: Yes, slide: No, push: Yes, Neotell: Yes

*** what Mary's capabilities are ***

1. Goal: Help me get the bolt up.

2. Position of objects: bolt: up: seen, door: shut: seen, zggoal: []: agreed, zgplan: []: agreed, zgexperiment: []: agreed, zgtest: []: agreed, zgask: []: agreed, zgtell: []: agreed, zggame: []: agreed, zgrule: []: agreed, John: out: seen, Mary: In: seen,

3. Consequences of events:
If you [[perform [robot slide]]] when [[is [any]]] then [[changes [undef]]].
If you [[perform [robot push]]] when [[is [bolt up]]] then [[changes [door]]].
If you [[perform[robot move]]] when [[is [any]]], then [[changes [nothing]]].
If you [[perform [robot slide]]] when [[cansee [bolt]]] then [[know [robot]]]
If you [[perform [robot push]]] when [[cansee [door]]] then [[know [bolt]]]
If you [[perform [robot move]]] when [[cansee [robot]]] then [[know [door]]]
If you [[perform [robot slide]]] when [[cando [bolt]]] then [[knowhow [bolt]]]
If you [[perform [robot push]]] when [[cando [door]]] then [[knowhow [door]]]
If you [[perform [robot move]]] when [[cando [robot]]] then [[knowhow [robot]]]
If you [[Neotell [Y]]] when [[Gricean [robot]]] then [[MutAgree [X]]]

4. Range of actions: move: Yes, slide: Yes, push: Yes, Neotell: Yes,

---

(State of the world is now
[John out, Mary in, bolt up, door shut])

1. John: I propose the following joint conversational goal: we agree whether the bolt is up.
2. Mary: OK
3. John: I propose you tell me if the bolt is up.
4. Mary: OK
5. Mary: I propose we find a test for seeing whether the bolt is up.
6. John: OK
7. John: I propose we experiment to see what happens when the door is pushed.
8. Mary: OK
9. Mary: John.
10. John: Yes.
11. Mary: I suggest we push the door and see if it opens.
12. John: OK
13. John: I suggest we plan for me to tell you about what happens when you push the door.
14. Mary: OK
15. John: Nothing happens.
16. Mary: I disagree.
17. Mary: I suggest we plan for me to tell you about what happens when you push the door.
18. John: OK
19. Mary: If you push the door when the bolt is up it changes position.
20. John: I see.
21. John: I suggest I push the door.
22. Mary: OK

(State of the world is now
[John out, Mary in, bolt up, door open])

23 John: I propose to tell you what has happened.
24 Mary: OK
25 John: I have pushed the door.
26 Mary: I see.
27. John: I propose to tell you the consequences of what happened when I pushed the door.
28. Mary: Go ahead.
29. John: The door changed position.
30. Mary: I see.
31. John: I propose the following joint conversational goal: we agree whether the door is open.
32. Mary: OK
33. John: I propose for me to tell you if the door is open.
34. Mary. OK
35. John: The door is now open.
36. John: I propose for you to tell me if the bolt is up.
37. Mary: OK
38. Mary: The bolt must be up.
39. John: Right.

Figure 6.2 Hypothetical dialogue illustrating how one generic procedure NEOTELL can generate all conversation for Power's robot world. Input parameters include a schema for causal rules about conversation (that contain variables X and Y) and the different kinds of physical planning (that instantiate X and Y).

Figure 6.2 illustrates a hypothetical dialogue with thirteen causal rules organised into four sets, one for each kind of planning. Each set then contains one rule for each of the three types of physical actions: pushing, moving and sliding. The preconditions and effects are defined in chapter 5. The thirteenth rule is the joint action for executing a piece of conversation. The goal is to mutually agree and the precondition is similar to Cohen and Perrault (1979). Thus, for all physical planning or inference there is a corresponding reasoning process for conversation that is instantiated in the same algorithm. Thus conversation can also be a goal of mutuality and that the equivalent action, solo mode, for mutually agreeing such a goal should be instigated before the conversational goal is considered. This could then be determined as a parameter of cooperation similar to Shadbolt's communicative postures that reflect the desired level of communication.

I now discuss the components of the rules given in figure 6.2.

6.3.3 The conversational rule.

This rule is composed of three parts:

If you do something, when such and such is true then the result is such and such

An action referred to as “evt” previously and “if you” in figure 6.2. A precondition referred to as “sit” previously and “when” in figure 6.2. An effect referred to as “res” previously and “then” in figure 6.2.

The middle “sit” component, GRICEAN (see footnote p81), which expresses the degree of co-operativeness or sensibility, is the precondition which must be true before the “evt” action component can be executed. For example, do not mutually agree with your partner about telling you some fact if you know he does not know it in the first place. More specifically, the preconditions for asking a question about getting your partner to do something are: do not plan for your partner to tell you something if you know he “can’t do it” (tell you it). Older systems have used different preconditions for telling and asking. Here, this is now no longer necessary with NEOTELL.

The action, NEOTELL, is structurally similar to a conversational procedure except most of the reasoning is done outside it, but nevertheless, still within the main physical planner. For example, conversational procedures such as GAME.ASSESS, Power (1979), are no longer needed because all the assessing is done in the planning stage. Similarly GAME.RULE is not needed since it is similar to telling an item and disagreements are handled as failed goals for which an alternative plan must be generated. Unlike in Power or Houghton, White utters an item for mutual agreement. Black does some goal recognition to identify the item and then evaluates in relation to its own knowledge. If it is okay, then the item is agreed. White then validates the reply. Both agents then take the item to be agreed and assess its consequences in just the same way as they would have assessed an action. The result is that the goal to mutually agree that particular item has been achieved. One advantage of doing things like this is that when mutually agreeing a goal the hearer has a better chance of taking the initiative in the dialogue. On the other hand, in the older systems, such as Power (1979) and Houghton (1986), agents are bound to reason either in a conversational procedure or within the planner. This disjointed reasoning process (i.e. partially reasoning within the physical planner and partially within the conversational procedure) means that agents are never able to contemplate whether or not to stay within a CP or IF. Mixed initiative joint planning, as illustrated in fig 6.2 and discussed in detail in chapter 5, although impossible with these systems is now easily achieved since facilities to process in parallel would be provided.

The "res" component (its effect) of NEOTELL is the original goal of talking about something that is treated just as any other type of physical goal would be. Just as the goal of, for example, opening the door must be planned for, so also does the goal of, for example, mutually agreeing to slide the bolt.

6.3.4 Physical rules.

The classical rules work in exactly the same way as described in Power (1979). The word "perform", is used here, see figure 6.2, as part of the action since in the planning phase the predicate means one thing (who can do it) and in the execution phase it means another (who or what has been done). The precondition for the instrumental predicate "cansee", see figure 6.2, is used to check whether or not the action can be assessed properly and is similar to Houghton's (1986) "get_info". The resulting predicate "know", see figure 6.2, means that the agent knows a particular fact after assessing the consequences of performing that action. In experimental planning the precondition predicate "cando", see figure 6.2, is similar to Houghton's (1986) interactional frame "gettodo", that is to say one needs to find an agent that can perform the action. Unlike Houghton these last two predicates do not make explicit calls to do further conversing but instead construct new goals to mutually agree items that the meta-planner directs control over (when appropriate) and the whole cycle begins again.

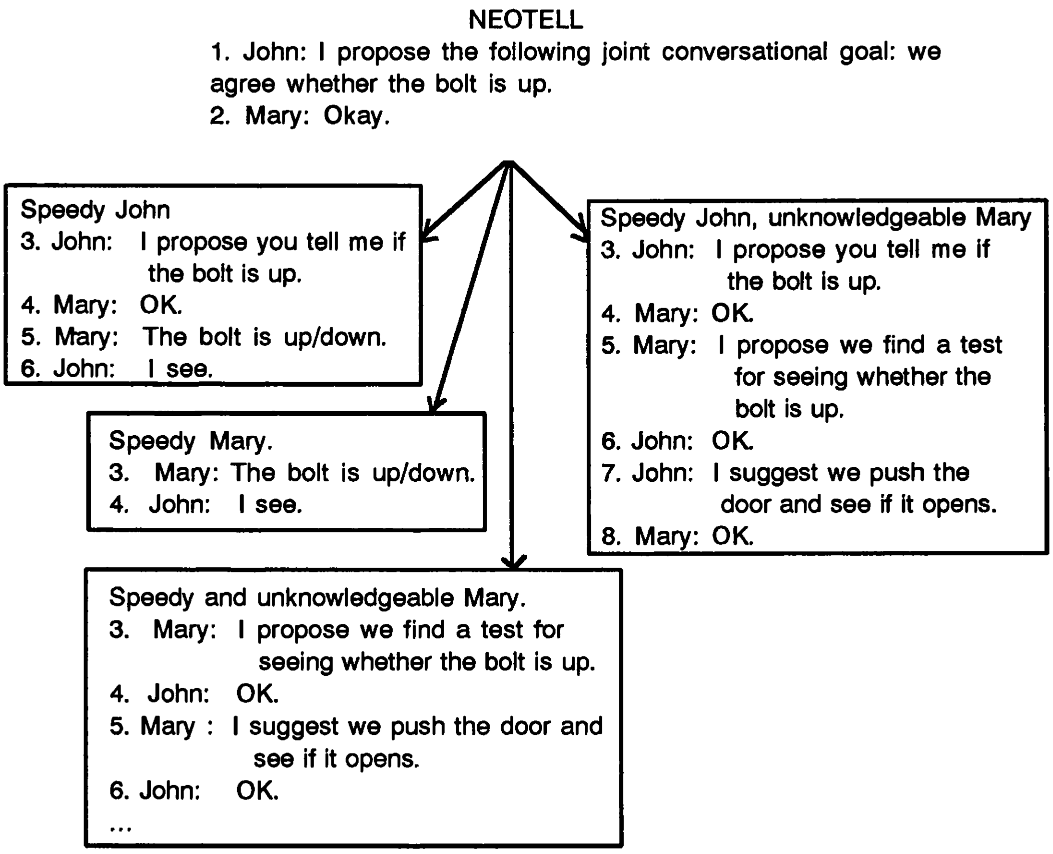
Even assessing the consequences of a piece of conversation is dealt with in the same way as with action. The task here, when a conversational exchange has taken place, is to assess its effects in exactly the same way as with an action, by comparing the old knowledge status with the new and updating accordingly. This is done within the inference and not the conversational procedure as currently implemented.

This also has the effect of providing a clearer definition, than the one mentioned in chapter 5, of what a topic at level 2 is, since the sub-goal is an item that needs to be mutually agreed and it closes when this goal has been achieved. In the definition in chapter 5 the topic closes when the conversational procedure finishes on the control stack. In this proposed representation, both the physical goal and the mutual goal now appear on the joint plan tree.

At first sight the hypothetical dialogue illustrated in figure 6.2 appears to be similar to Shadbolt's dialogue illustrated in figure 3.5 in that it is almost efficient (like Houghton's dialogue), certainly regular and mechanistic in appearance and definitely more explicit than even Power. There are however some differences. The dialogue is more robust for two reasons: firstly either agent can seize the initiative at any time during execution (chapter 4), secondly if an agent is in difficulties alternative plans can be constructed whether or not the agent's partner is being helpful (chapter 5).

6.3.5 Asking a question with NEOTELL.

The following tree diagram (figure 6.3) shows how it would be possible to ask a question with NEOTELL. In every



NEOTELL
1. John: I propose the following joint conversational goal: we agree whether the bolt is up.
2. Mary: Okay.

Speedy John
3. John: I propose you tell me if the bolt is up.
4. Mary: OK.
5. Mary: The bolt is up/down.
6. John: I see.

Speedy Mary.
3. Mary: The bolt is up/down.
4. John: I see.

Speedy John, unknowledgeable Mary
3. John: I propose you tell me if the bolt is up.
4. Mary: OK.
5. Mary: I propose we find a test for seeing whether the bolt is up.
6. John: OK.
7. John: I suggest we push the door and see if it opens.
8. Mary: OK.

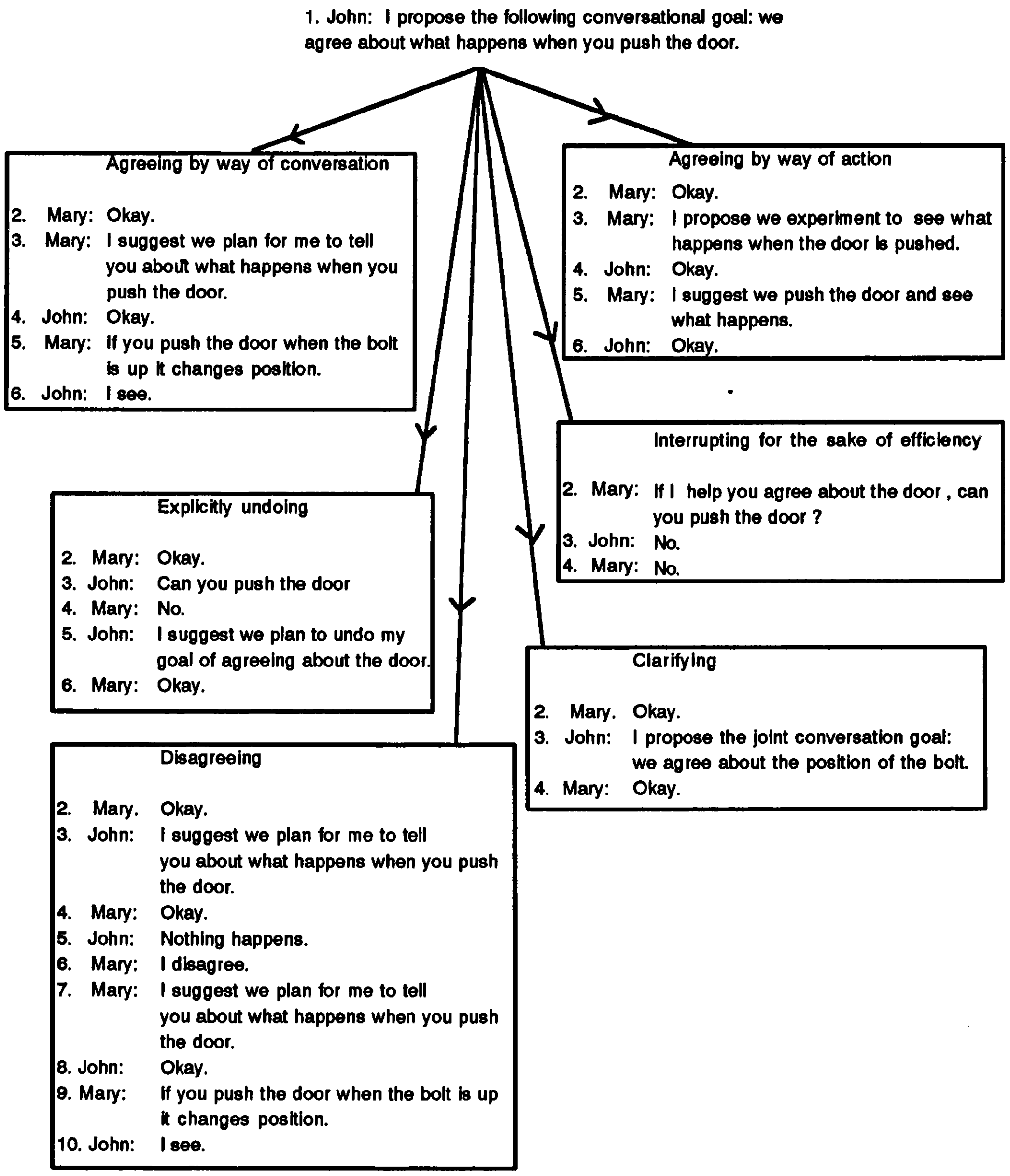
Speedy and unknowledgeable Mary.
3. Mary: I propose we find a test for seeing whether the bolt is up.
4. John: OK.
5. Mary : I suggest we push the door and see if it opens.
6. John: OK.
...

Figure 6.3 Asking the question "Is the bolt up?" using NEOTELL.

situation the agent can plan or discuss depending on the conversational skills that are applied as input parameters.

6.3.6 Degrees of conflict resolution with NEOTELL.

Similar diagrams to figure 6.3 can also be constructed for inferring the consequences of an action and for degrees of conflict resolution (figure 6.4).


1. John: I propose the following conversational goal: we agree about what happens when you push the door.

Agreeing by way of conversation

2. Mary: Okay.
3. Mary: I suggest we plan for me to tell you about what happens when you push the door.
4. John: Okay.
5. Mary: If you push the door when the bolt is up it changes position.
6. John: I see.

Agreeing by way of action

2. Mary: Okay.
3. Mary: I propose we experiment to see what happens when the door is pushed.
4. John: Okay.
5. Mary: I suggest we push the door and see what happens.
6. John: Okay.

Interrupting for the sake of efficiency

2. Mary: If I help you agree about the door , can you push the door ?
3. John: No.
4. Mary: No.

Explicitly undoing

2. Mary: Okay.
3. John: Can you push the door
4. Mary: No.
5. John: I suggest we plan to undo my goal of agreeing about the door.
6. Mary: Okay.

Clarifying

2. Mary. Okay.
3. John: I propose the joint conversation goal: we agree about the position of the bolt.
4. Mary: Okay.

Disagreeing

2. Mary. Okay.
3. John: I suggest we plan for me to tell you about what happens when you push the door.
4. Mary: Okay.
5. John: Nothing happens.
6. Mary: I disagree.
7. Mary: I suggest we plan for me to tell you about what happens when you push the door.
8. John: Okay.
9. Mary: If you push the door when the bolt is up it changes position.
10. John: I see.

Figure 6.4 Degrees of Conflict resolution with NEOTELL.

6.3.7 Meta-planning with NEOTELL.

John is out and Mary is in. The bolt is up and the door is shut. Mary is speedy relative to John. Mary and John's top goal is for the door to be open which has already been mutually agreed by both participants. John's current sub-goal is to know whether or not the bolt is up since he is out and cannot see it. Mary's current sub-goal is to push the door. Before Mary can push the door John says "I suggest we mutually agree a goal to find out if the bolt is up?" How should Mary reply?

As mentioned earlier a meta-planner is needed to plan about what to plan for next. A meta-inference process is also needed to control the cases of multiple inferences. It is here that the satisfaction of constraints would be appropriate. Since in this case the meta-planner would be faced with a number of simultaneous alternatives. Mary can do two things. She can either mutually agree with John's goal or she can make a meta-action response by pushing the door. This not only achieves her own goal but also indirectly indicates to John that the bolt is up since the result of the door being open is that the bolt must be up.

Hence the corresponding action for mutually agreeing is conveying the message through direct action which may involve miming, a planned action, acting in solo mode or any other form of non-verbal communication. But whatever the conversation, from a psychological point of view and if this model is upheld, there is always a corresponding action for achieving it.

Thus the responsibility of the meta-planner is for efficiency (see also the example in figure 6.4 “interrupting for the sake of efficiency”) and this can be achieved through constraint programming as part of the existing inference mechanism already implemented within SUPERPOWER.

6.4 Epilogue.

This NEOTELL dialogue is slightly more general than the dialogue of chapters 4 and 5 in that a fast agent can attempt to answer a question that it is pondering by way of its own resources before its partner has replied. It thus introduces even more complications to exercise the agent's abilities to coordinate skills cooperatively. Thus the variation and distribution of conversational procedures, skills and speeds amongst agents is even larger than indicated in chapter 4. It also begs the question “For a given level of overall skill and an arbitrary selection of goals and unexpected events, how do the relative speeds and the configuration of skills amongst the agents affect the dialogue outcome in terms of the distribution of conversational procedures instigated by each agent?”. In

Chapter 4, there were some initial conclusions about this, but much more information still needs to be gathered on this. With this proposed enhancement the number of shifts in which an agent is leading the dialogue would be much greater.

There is also the general issue about equivalent actions for mutually agreeing an item. Does every utterance have a way of being expressed in terms of some action? Dumb people may agree. Sometimes a glance of an action can represent a lifetime's insight. With other important insights it can be just one word. The computational design described in this chapter is an attempt at addressing this important psychological question of whether or not every piece of conversation can be expressed in terms of an action and vice versa.

By way of application to Cognitive Science, this is also a question that is important for understanding human transcripts (Anderson, Clark and Mullin 1989) by helping us to identify a model of true planned conversation we can identify poor and good interaction skills. It also has application both in intelligent language tutoring systems (see Button and Draper 1990) that use communicative teaching strategies, (see Ward 1989 for some empirical work that justifies this approach to language teaching). It may even be of interest to Psycholinguists. It has application to Psychology since any

Social theory of mutual intention must explain its corresponding action and up until now these problems have been dealt with differently.

It can be applied to constraint satisfaction since conversational and physical goals are expressed in such a way as to make them amenable to reasoning in relation to the constraints of the overall situation. It is at this point that meta-planning is useful to make the dialogue efficient. Goals do not need to be specified as explicitly as in Power's model, since conversational exchanges can be treated just like unexpected actions and assessed accordingly. The system is thus reactive to the current situation. The task of the inference process is to use constraint programming as a way of resolving goal conflict. The inference and the meta-planner with this system would thus be the "constraint programmer" mentioned in chapter 2. It's just that we need to define an explicit program that can say all that there is to say for dialogue structure first.

However, a process that carries forward multiple goals, and progressively disambiguates meaning by making contributions that add new constraints to the understanding of a developing plan, while easily accomplished by human conversation partners, takes us far beyond any simulation of conversational procedures that has been implemented so far. However, I have also argued that an explicit account of simple levels of goals is technically possible and is the way forward for any AI model of dialogue that wishes to account for true planned conversation and action, as opposed to simply understanding planned action and conversation separately. It must, at all times, follow principles of coherence in terms of theory, program structure and example dialogue. And when the possibilities become too large, comprehensive statistical summaries must also be made.

References.

Agre, P.E., (1990). Book Review of Suchman (1987). Artificial Intelligence, 43, 369-384.

Allen, J.F., (1983). Recognising intentions from natural language utterances. In Computational Models of Discourse, (Eds.), M. Brady and R.C. Berwick, 107-166. Cambridge, MA: MIT Press.

Allen, J., (1987). Natural Language Understanding. Menlo Park: Benjamin Cummings.

Allen, J.F. and Perrault, C.R., (1980). Analysing intention in utterances. Artificial Intelligence, 15(3), 143-178. Reprinted in Readings in Natural Language Processing, (Eds.) B.J. Grosz, K. Sparck-Jones and B.L. Webber (1986), 441-458. Los Altos, CA: Morgan Kaufmann.

Anderson, A.S., Clark, A., and Mullin, J., (1989). The development of referential communication skills: interactions between speakers and listeners in extended dialogues. Paper presented at the 3rd E.A.R.L.I. Conference, Madrid 4-7th Sept.

Appelt, D.E., (1985). Planning English Sentences. Cambridge: Cambridge University Press.

Appelt, D.E., (1987). Towards a plan-based theory of referring actions. In Natural Language Generation, pp63-70, (Ed.), G. Kempen. Also in Proc. Third International Workshop on Natural Language Generation, Aug. 12-23 1986. Netherlands: Martinus Nijhoff.

Austin, J.L., (1962). How to do things with words. Oxford: Oxford University Press.

Austin, J.L., (1970). Philosophical papers. Oxford: Clarendon Press. Also reprinted from Proc. Aristotelian Society, suppl. xxxii, (1957-1958).

Baddley, A.D., (1976). The Psychology of memory. New York: Basic books.

Bartlett, F.C., (1932). Remembering: A study in Experimental and Social Psychology. Cambridge, England: University Press.

Bennet, A., (1981). Interruptions and the interpretation of discourse. Discourse Processes, 4, 177-188.

Bobrow, D.G., Kaplan, R.M., Kay, M., Norman, D.A., Thompson, H., and Winograd, T., (1976). GUS, A frame-driven Dialogue system. Artificial Intelligence, 8, 155-173.

Bratman, M., (1987). Intentions, Plans and Practical reason. Cambridge, MA: Harvard University Press.

Brown, F.M., (1987). Procs. of the 1987 workshop on the Frame problem in Artificial Intelligence. Los Altos, CA: Morgan Kaufmann.

Bull, P. and Mayer, K., (1988). Interruptions in Political Interviews: A study of Margaret Thatcher and Neil Kinnock. Paper presented to the B.P.S, Leeds, 15-18 April.

Butterworth. B.L., (1980). Language Production, Vol. 1: Speech and Talk. London: Academic Press.

Button, C.G., (1989). An intelligent tutoring system for helping children aged 7 to 15 to acquire a second language. Instructional Science, 18(1), 27-43.

Button, C.G., Oatley, K. and Draper, S.W., (1989). Applying features of purposeful conversation to an intelligent tutoring system for children's acquisition of a second language. Cognitive Systems, 2(3), 261-273.

Button, C.G. and Draper, S.W., (1990). Towards a computational theory of the constituents in a unit of purposeful conversation. Paper presented at the NATO sponsored workshop on Intelligent Language tutoring systems, Washington D.C., Sept 1990.

Carletta, J., (1990). A general architecture for Interactive Explanations. Unpublished technical report, Department of Artificial Intelligence, University of Edinburgh.

Charniak, E. and McDermott, D.V., (1985). Introduction to Artificial Intelligence. Reading, MA: Addison Wesley.

Clark, H.H., (1985). Language use and language users. In G. Lindsey and E. Aronson, The Handbook of Social Psychology (3rd edition, Vol. 2), 179-231. Hillsdale, NJ: Lawrence Erlbaum and Associates.

Clark, H.H. and Schaefer, E.F., (1987). Collaborating on contributions to conversations. Language and Cognitive Processes, 2,19-41.

Clocksin, W.F. and Mellish, C.S., (1981). Programming in Prolog. Berlin: Springer-Verlag.

Coates, J., (1990). Modal Meaning: The Semantic-Pragmatic Interface. Journal of Semantics, 7, 53-63.

Cohen, P.R. and Levesque, H.J., (1985). Speech acts and rationality. A.C.L. Procs., 23rd Annual Meeting, 49-60.

Cohen, P.R. and Levesque, H.J., (1990). Intention is Choice with Commitment. Artificial Intelligence, 42, 213-261.

Cohen, P.R. and Perrault, C.R., (1979a). Elements of a plan-based theory of speech acts. In Readings in Natural Language Processing, (Ed.), B.J. Grosz, K. Sparck-Jones and B.L. Webber (1986), 423-440. Los Altos, CA: Morgan Kaufmann.

Cohen, P.R. and Perrault, C.R., (1979b). Elements of a plan based theory of speech acts. Cognitive Science, 3, 177-212.

Collins, A. M. and Quillian, M.R., (1969). Retrieval time from semantic memory. Journal of Verbal Learning and Verbal Behaviour, 8, 240-247.

Crowder, R.G., (1982). The demise of short-term memory. Acta Psychologica, 50, 291-323.

Davey, A., (1978). Discourse Production: A computer model of some aspects of a speaker. Edinburgh: Edinburgh University Press.

Draper, S.W. and Button, C.G., (1990). Conversation as planned action: planning utterances within dialogue. NATO sponsored workshop, Trento Italy, November 1990.

Draper, S.W., Oatley, K. and Garrod, S.C., (1987). 4 goal levels - designing the research. Unpublished Mimeo, Department of Psychology, University of Glasgow.

Elsom-Cook, M.T. (1984). Design considerations of an intelligent tutoring system for programming languages. Unpublished PhD thesis, University of Warwick.

Fikes, R. and Nilsson, N.J., (1971). STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2, 189-208.

Freidland, P.E. and Iwasaki, Y., (1985). The concept and implementation of skeletal plans. Journal of Automated Reasoning, 1, 161-208.

Freud, S., (1904). The psychopathology of everyday life. In J. Strachey et al. (Eds.) The standard edition of the complete psychological works of Sigmund Freud, Vol. 4. London: Hogarth Press. Also in the Institute of Psychoanalysis, 1953, pp. 107-118.

Galliers, J.R., (1987). Modelling Dialogue involving conflict and co-operation. Working Paper H.C.R.L., Open University.

Garfinkel, H., (1967). Studies of the routine grounds of everyday activities. In H. Garfinkel, (Ed.), Studies in ethnomethodology. Englewood Cliffs, NJ: Prentice Hall.

Garnham, A., (1985). Psycholinguistics, central topics. London: Mentuen.

Garrod, S. and Anderson, A., (1987). Saying what you mean in dialogue: a study in conceptual and semantic coordination. Cognition, 27, 181-218.

Garrod, S. and Sanford, A.J., (1985). Thematic subjecthood and cognitive constraints on discourse structure. Journal of Pragmatics, 12, 357-372.

Gazdar, G. and Mellish, C., (1990). Natural Language Processing in POP11. An introduction to computational linguistics. Wokingham, England: Addison-Wesley.

Georgeff, M.P., (1987). Actions, processes and causality. In M.P Georgeff and A.L. Lansky, (Eds.), Reasoning about actions and plans. 99-122 Proc.1986 Workshop. Los Altos, CA: Morgan Kaufmann.

Ginsberg, M.L. and Smith, D.E., (1986). Reasoning about action II: the Qualification problem. Technical Report 86-66, KSL, Stanford University. Also in M.P. Georgeff and A.L. Lansky, (Eds.), Reasoning about actions and plans Procs. of 1986 Workshop.1987. Los Altos, CA: Morgan Kaufmann. Also in Artificial Intelligence, 35(3), 1988.

Grice, H.P., (1957). Meaning. Philosophical Review, 66, 377-388.

Grice, H.P., (1968). Utterers meaning, sentence meaning and word meaning. In J. Searle, (Ed.), The Philosophy of Language, pp 1-18. London: Oxford University Press. Also printed in Foundations of language, 4, 225-242.

Grice, H.P., (1975). Logic and conversation. In P. Cole and J.L. Morgan, (Eds.), Syntax and semantics 3: Speech acts, 41-58. New York: Academic Press.

Grosz, B., (1977). The representation and use of focus for understanding dialogs. Proc. fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, pp 67-76. Los Altos, CA: Morgan Kaufmann.

Grosz, B.J. and Sidner, C.L., (1986). Attentions, intentions, and the structure of discourse. Computational Linguistics,12(3), 175-204.

Grosz, B.J., Sparck-Jones, K., and Webber, B.L., (1986). Readings in Natural Language Processing. Los Altos, CA:

Morgan Kaufmann.

Halliday, M.A.K., (1961). Categories of the theory of grammar. Word, 17.

Halliday, M.A.K., and Hasan, R., (1976). Cohesion in English. London: Longman.

Harley, T.A., (1984). A critique of Top-down Independent levels Models of Speech Production: Evidence from Non-plan Internal Speech Errors. Cognitive Science, 8,191-219.

Hirst, G., (1981). Anaphora in natural language understanding, a survey. Lecture notes in Computer Science, Vol. 119. Berlin and New York: Springer Verlag.

Hobbs, J. and Evans, D., (1980). Conversation as planned behaviour. Cognitive Science, 4, 349-377.

Houghton, G., (1986). The production of language in dialogue: a computational study. Ph.D. thesis, University of Sussex.

Houghton, G. and Isard S., (1987). Why to speak, What to say and How to say it : modelling language production in discourse. In P.Morris, (Ed.), Modelling Cognition. Chichester: Wiley.

Houghton, G., (1989). A Computational Model of Discourse Production. Norwood, NJ: Ablex.

Jung, C.G., (1910). The association method. In H. Read et al, (Eds.), The collected works of C.G. Jung. Vol. 2. Experimental Researches, (1973), 439-465. Princeton, N.J.: Princeton University Press.

Lansky, A.L., (1986). A representation of parallel activity based on events, structure and causality. Technical note 401. December 1986, SRI International.

Lansky, A.L., (1988). Localised event-based reasoning for multiagent domains. Technical note 423. January 1988, SRI International.

Litman, D.J. and Allen J.F., (1987). A plan recognition model for subdialogues in conversations. Cognitive Science, 11, 163-200.

Musson, C. and Shadbolt, N., (1987). A summary and comparison of the work of J.F. Allen and R. Power High-level dialogue. Technical Report, Department of Psychology, University of Nottingham.

Oatley, K., Draper, S.W. and Button, C.G., (in press). Goals in conversation: Increments of agreement and the satisfaction of constraints. Unpublished report, Dept. Psychology, University of Glasgow.

Palmer, F.R., (1990). Modality and the English Modals. Second Edition, London-Longman.

Parkison, R.C., Colby, K.M., and Faught, W.S., (1977). Conversational Language Comprehension Using Integrated Pattern-Matching and Parsing, Artificial Intelligence, 9, 111-134.

Pollack, M.E., (1986). Inferring Domain Plans in Question-Answering. Ph.D. thesis, University of Pennsylvania.

Power, R.J.D., (1974). A Computer model of Conversation. Ph.D. thesis, University of Edinburgh.

Power, R.J.D., (1979). The organisation of purposeful dialogues. Linguistics, 17(1),107-152.

Power, R.J.D., (1984). Mutual Intention. Journal for the Theory of Social Behaviour, 14, 85-102.

Power, R.J.D., (1987). Efficiency in Conversation. Alvey Workshop on Explanation, Surrey University, January 8-9, 1987.

Power, R.J.D. and Dal Martello, M.F., (1986). Some criticisms of Sacks, Schegloff, and Jefferson on turn taking. Semiotica, 58, 29-40.

Reichman, R., (1978). Conversational Coherency. Cognitive Science, 2, 283-327.

Reichman, R., (1986). Communication paradigms for a window system. In User Centred System Design, (Eds.), Norman, D. and Draper, S. London: Lawrence Erlbaum and Associates.

Rogers, C., (1951). Client Centred Therapy: Current Practice, Implications and Theory. Boston, Houghton Mifflin.

Sacerdoti, E.D., (1974). Planning in a hierarchy of abstraction spaces. Artificial Intelligence, 5, 115-135.

Sacks, E., Schegloff, E. and Jefferson, G., (1974). A simplest systematics for the organisation of turn taking in conversation. Language, 50, 696-735.

Sanford, A.J. and Garrod, S.C., (1981). Understanding Written Language. Chichester: Wiley.

Schank, R.C., (1972). Conceptual dependency: a theory of natural language understanding. Cognitive Psychology, 3, 552-631.

Schank, R.C. and Abelson, R., (1977). Scripts, Plans, Goals and Understanding. Hillsdale, NJ: Lawrence Erlbaum and Associates.

Schank, R.C., (1980). Language and Memory. Cognitive Science, 4, 243-284.

Schank, R.C. and Reisbeck, C.K., (1981). Inside Computer Understanding. Hillsdale, N J: Laurance Erlbaum and Associates.

Schank, R.C., (1982). Dynamic Memory: A theory of Learning in Computers and People. Cambridge: Cambridge University Press.

Schank, R.C., (1985). Reminding and Memory Organisation. In A.M. Aitkenhead and J.M. Slack, (Eds.), Issues in Cognitive modelling. Hillsdale, NJ: Lawrence Erlbaum and Associates.

Searle, J.R., (1969). Speech acts: An essay in the philosophy of language. Cambridge: Cambridge University Press.

Searle, J.R., (1975). Indirect speech acts. In P. Cole and J. Morgan, (Eds.), Syntax and semantics, Vol. 3: Speech acts, pp 59-82. New York: Academic Press.

Searle, J.R., (1983). Intentionality: An Essay in the Philosophy of Mind. New York: Cambridge University Press.

Searle, J.R., (1990). Collective Intentionality. In P.R. Cohen,

J. Morgan and M.E. Pollack, (Eds.), Intentions in Communication. Cambridge, MA: MIT Press.

Shadbolt, N.R., (1984). Constituting Reference in Natural Language: the problem of referential opacity. Ph.D. thesis, University of Edinburgh.

Shadbolt, N.R. and Musson C.L.D.L., (1987). co-operative Planning: a foundation for communicative negotiation. Procs. of the 6th Alvey SIGPLAN meeting, Cambridge.

Steedman, M. and Johnson-Laird, P., (1980). The Production of Sentences, Utterances and Speech Acts: Have Computers anything to say? In B.L. Butterworth, (Ed.), Language Production, Vol. 1: Speech and Talk. London: Academic Press.

Steel, S., (1987). The bread and butter of planning. AI Review, 1, 159-181.

Stefik, M., (1980). Planning with constraints. Stanford University Computer Science Department Technical Report.

Stefik, M., (1981). Planning and meta-planning, MOLGEN: part 2. Artificial Intelligence, 12, 141-169.

Stubbs, M., (1986). A matter of prolonged fieldwork: notes towards a modal grammar of English. Applied linguistics, 7, 1-25.

Suchman, L.A., (1987). Plans and situated actions: the problem of human machine communication. Cambridge: Cambridge University Press.

Tate, A., (1976). Project planning using a hierarchic non-linear planner. Dept. AI memo no 25, University of Edinburgh.

Tate, A., (1985). A review of Knowledge-Based Planning techniques. Knowledge Engineer's Review, Vol 1, No 2, 1985.

Turner, R., (1984). Logics for Artificial Intelligence. Chichester: Ellis Horwood.

Ward R.D., (1989). Some uses of natural language interfaces in computer assisted language learning. Instructional Science, 18: 45-61.

Webber, B. L., (1987). Question answering. In Encyclopaedia of Artificial Intelligence, (Ed.), S. C. Shapiro, pp. 814-822. New York: Wiley.

Weizenbaum, J., (1966). ELIZA - a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36-45.

Wilensky, R., (1983). Planning and Understanding. Reading Mass: Addison Wesley.

Wilkins, D.E., (1984). Domain-Independent Planning: Representation and Plan Generation. Artificial Intelligence, 22, 269-303.

Winograd, T., (1972). Understanding Natural Language. New York: Academic Press.

Wood, S., (1990). Planning in a rapidly changing environment. Ph.D. thesis, University of Sussex.

Appendix 1.

Annotated transcript of task of getting through a closed door (conversation by humans, one blindfolded).

	Agent	Utterance	G/P/B/H/C	Action	Change
		John and Mary decide to gain each other's attention in order to achieve John's goal of getting into Mary's room.			
1	JOHN:	[Mary.			
2	MARY:	Yeh.]	H		
		John starts by executing a plan, sliding the bolt, and then asking Mary to assist him achieve his goal of getting in by pushing on the door. He sees it is not working, and asks why.			
				JOHN OPENS BOLT	BOLT OPEN
3	JOHN:	[Can you push on the door?			
4	MARY:	[Can I what?			
5	JOHN:	Can you push on the door, Mary?]	H	MARY PUSHES DOOR	NO CHANGE
6	MARY:	[Push the door?			
7	JOHN:	Yah...]	H		
		]	P	MARY PUSHES DOOR	NO CHANGE
		[Keep pushing.			
		]	P	MARY PUSHES DOOR	NO CHANGE
		[Is anything wrong?			
8	MARY:	I can't get through.]	B		
		John modifies his plan, then tries to find out what is wrong by introducing new knowledge.			
9	JOHN:	[Well, try pushing a bit lower.			
		]	P	MARY PUSHES DOOR	NO CHANGE
		[Is it stuck?			
10	MARY:	Yeh.]	B		
		They compare their beliefs about the state of the world, and conclude the door is jammed.			

Agent	Utterance	G/P/B/H/C	Action	Change
11 JOHN:	[Where do you think it's stuck?			
12 MARY:	[Where I think it's stuck?			
13 JOHN:	Yeh...]	H		
	[Is it stuck down the bottom?			
14 MARY:	...According to the crack obviously it's not locked.]	B		
	]	B		
	They continue to discuss beliefs about appearances on either side of the door.			
15 JOHN:	[No it's not locked. I've got it open this side. But it's stuck somewhere.			
16 MARY:	Yeah I think it's the bottom because the upper part is free.]	B		
17 JOHN:	[Can you give it a push from the bottom...			
	]	P	MARY PUSHES DOOR	NO CHANGE
	[It's actually stuck where the lock is, about 2 inches below the Yale lock... About 6 inches below.			
18 MARY:	So?]	B		
19 JOHN:	[Have you got a handle your side?			
20 MARY:	A handle! No.]	B		
21 JOHN:	[My handle's undone as well.			
	]	B		
	Mary is at a loss as to what to do. John continues to try and clarify things. Finally they agree that it is not Mary who needs to do something but John.			
22 MARY:	[What shall I do?			
23 JOHN:	[What can you see from your side?			
24 MARY:	See? I'm not supposed to see anything.]	B		
	Mary was blindfolded!			
25 JOHN:	[Okay. Can you see anything that's stopping it?			
26 MARY:	Stopping it! Obviously you're stopping it not me.]	B		
	]	P		
	Finally John stumbles on the solution. He modifies his beliefs and plans to achieve his goal of getting in by removing the hidden nail from the side of the door.			
27 JOHN:	[Oh wait a minute there's a nail in the side.]	B	JOHN REMOVES NAIL	
			MARY PUSHES DOOR	DOOR OPEN
			JOHN MOVES	JOHN IN

Appendix 2.

Annotated transcript of double-sided photocopying task (conversation by two humans).

Agent	Utterance	G/P/B/H/C	Action	Change
	John and Mary start conversing about photocopying both sides of a sheet of paper. Mary knows how to do this. John's goal is to learn how to do it for himself. After the first few utterances it emerges that John knows how to do one-sided photocopying. Mary tries to guide John from beliefs about single-sided photocopying to beliefs about double-sided photocopying which involve topological problems about the orientation and sequencing of originals. Further complications arise because Mary is unaware that John only intends to photocopy one sheet rather than many ...			
1 JOHN:	[Could you show me how to do the photocopying?			
2 MARY:	[Double-sided?			
3 JOHN:	Eh, Yer, I want to do, to do double-sided.]	G		
	]	G		
4 MARY:	[Uhm, I don't know, some or]	B		
	John interrupts Mary's goal-clarification. He uses his beliefs about single-sided photocopying to develop a plan to achieve his goal, with an additional unstated sub-goal of doing the task as quickly as possible. He knows that of four keys, three are relevant to single-sided photocopying and infers that the fourth must concern double-sided photocopying...			
5 JOHN:	[Sorry, what do you do here?			
6 MARY:	This one.		SELECTS DUPLEX 2	DUPLEX SCREENAT2
	But eh eh... some turn the other way round. You must have it.]	C:B,P	OPENS LID	LID OPEN
	Mary has in mind that double-sided copying is not as simple as single-sided, because documents to be copied double-sided must be in sequence.			
7 JOHN:	[So what do I do just put that one there? Like that.		POSITIONS PAPER	PAPER ON GLASS
8 MARY:	]	P	SETS KEY TO A4	SET TO A4
	Mary checks that the first sheet is in position. John is now confused as to why Mary is so concerned about the sequencing of the originals.			
10 MARY:	[Which way did you put it?			
11 JOHN:	It doesn't really matter which way you put it.]	C:P,B		
	[Does it?			
12 MARY:	Yer, it must be in sequence.]	B		
	[I suppose if you...			
13 JOHN:	Right well, I'm only doing one sheet.]	C:B,P		
	Mary is at last able to understand more of John's goals and that the confusion between them over-sequencing is not important when photocopying one sheet of paper. However she is still perplexed as to why someone should go to great lengths to photocopy just one sheet of paper double-sided.			

Agent	Utterance	G/P/B/H	Action	Change
14 MARY:	[Pardon?			
15 JOHN:	It's alright. I'm just learning how to do it at the moment.			
16 MARY	Oh I see.]	G		
	Now the rest is straightforward for both of them.			
17 JOHN:	[Right, so I put that down there. Right?			
18 MARY:	ahm...		CLOSES LID	LID CLOSED
19 JOHN:	And press that do I?			
20 MARY:	Yes.]	P	PRESSES START	MACHINE STARTS
	[Then it collects...			ONE SIDED COPY APPEARSBELOW
	and goes back you see.]	B		
	[Now now you turn it.]	P		
	Having successfully completed a photocopy of the first side, John hastily prepares to press the start button again for the reverse side...			
21 JOHN:	[What I press again?		POINTS TO START	
22 MARY:	No, no. Well... ha ha ha.]	P		
23 JOHN:	[I have to turn it over,			
	do I, first.		OPENS LID TURNS SHEET	LID OPEN SHEETTURNED
	Right.			
24 MARY:	Ha Ha.]	P		
	Mary wants to make sure that when John turns the sheet over he puts it the right way round to avoid having the reverse side copied upside down.			
24 MARY:	[So it might be you have to turn it the other way round but I'm not sure.			
25 JOHN:	Oh. What you mean we might get it the wrong			
	way up. Right.]	P	POSITIONS PAPER CLOSES LID	PAPER ON GLASS LID CLOSED
	[So I press it again.			
26 MARY:	Yes.]	P	PRESSES START	MACHINE STARTS
27 JOHN:	[... Right.]	P		
28 MARY:	[So it comes out here.			DOUBLE-SIDEDCOPY APPEARS AT TOP
29 JOHN:	Sorry. It comes out where,			
	Sorry?]	B	MARY POINTS	
30 MARY:	[It takes much longer.]	B		

Appendix 3.

Utterance in the photocopying conversation, categories assigned by the experimenters, the number of subjects assigning each utterance to each category, and the percentage agreement of subjects with our categories.

Agent	Utterance	Cat	G	P	H	B	C:B,P	O	%Agrmt
JOHN:	[Could you show me how to do the photocopying?								
MARY:	-]	G	17	1	0	0	0	0	94
MARY:	[Double-sided?								
JOHN:	Er, Yer, I want to do,to do, double sided]								
		G	7	6	2	0	0	3	44
MARY:	[I don't know, some or ...								
JOHN:	-]	B	0	1	8	6	0	3	33
JOHN:	[Sorry, what do you do here?								
MARY:	This one. But eh , eh... some turn the other way round.								
	You must have it.]	C:P,B	0	8	0	2	5	3	64
JOHN:	[So what do I do just put that one there? Like that.								
MARY:	-]	P	0	9	1	6	0	2	53
MARY:	[Which way did you put it?								
JOHN:	It doesn't really matter which way you put it.]								
		C:P,B	0	1	1	11	2	3	50
JOHN:	[Does it ?								
MARY:	Yes it must be in								
	sequence.]	B	0	6	0	8	0	4	56
MARY:	[I suppose if you...								
JOHN:	Right, well I'm only								
	doing one sheet.]	C:B,P	1	1	2	4	3	7	42
MARY:	[Pardon?								
JOHN:	It's alright I'm just learning how to do it at the moment.								
MARY:	Oh I see.]	G	1	0	9	0	0	8	6
JOHN:	[Right, so I put that down there. Right?								
MARY:	ahm								
JOHN:	And press that, do I?								
MARY:	Yes]	P	0	10	1	5	1	1	58
MARY:	[Then it collects down here and goes back you see.]								
		B	0	2	0	13	0	3	72
MARY:	[Now you turn it.]	P	0	11	3	3	1	0	64
JOHN:	[What I press again.								
MARY:	No. Ha ha ha.]	P	0	5	1	4	4	4	44
JOHN:	[I have to turn over again first. Right.								
MARY:	Ha Ha.]	P	0	11	0	4	0	3	61
MARY:	[So it might be you have to turn it the other way round, but I'm not sure.								
JOHN:	Oh you mean we might get it the wrong way up.								
	Right.]	P	0	3	0	5	9	1	44
JOHN:	[So I press it again.								
MARY:	Yes]	P	0	10	1	3	4	0	67
JOHN:	[Right.]	P	0	1	10	2	0	5	6
MARY:	[So it comes out here.								
JOHN:	Sorry. It comes out where,								
	Sorry.]	B	0	2	0	12	1	3	69
MARY:	[It takes much								
	longer.]	B	0	1	5	7	1	4	42
	Totals		26	89	44	95	31	57	

Notes: Cat = categories agreed by the authors; G,P,B,H, & C:B,P = coding categories assigned by subjects as described in the text; O = coding categories assigned by subjects other than the foregoing; %Agrmt = percentage agreement between subjects' categorisations and the authors', with half points being given for a partially correct assignment, e.g. where we assigned a single category and a subject assigned a compound containing this category.

Appendix 4.

An attached list of dialogue summaries for the goal of agent John getting the door open. Each record contains a list of input and output parameters of the dialogue.

<-------MIX OF SKILLS--->						<-------MIX OF SPEEDS--->											
Knowledge		Action		Perception		Total CP's				CP's by John				CP's by Mary			
						SF	FS	SS	FF	SF	FS	SS	FF	SF	FS	SS	FF
KNW	KNW	PRA	PRA	PER	PER	10	10	10	10	2*	10*	6	6	8*	0	4	4
Skill level = 6 Av # of CPs						10	10	10	10	2	10	6	6	8	0	4	4
KNW	UNK	PRA	PRA	PER	PER	11	10	11	11	5	10*	5	5	6*	0	6*	6*
KNW	KNW	PRA	IMPRA	PER	PER	12	10$	12	12	4*	10*	4*	8	8*	0	8*	4
KNW	KNW	PRA	PRA	PER	IMPER	19$	15	14	14	6	9	8	8	13*	6	6	6
UNK	KNW	PRA	PRA	PER	PER	10	11	10	10	2*	8*	6	6	8*	3*	4	4
KNW	KNW	IMPRA	PRA	PER	PER	10	12$	10	10	2*	10*	6	6	8*	2*	4	4
KNW	KNW	PRA	PRA	IMPER	PER	15	14	15	14	8	8	8	8	7	6	7	6
Skill level = 5 Av # of CPs						12	13	12	11	4	9	6	6	8	4	5	5
KNW	UNK	PRA	IMPRA	PER	PER	11	10	11	11	5	10*	5	5	6*	0	6*	6*
KNW	UNK	PRA	PRA	PER	IMPER	13	13	12	12	7	9	8	8	6*	4	4	4
KNW	KNW	PRA	IMPRA	PER	IMPER	16	15	14	14	6	9	8	8	10*	6	6	6
UNK	KNW	IMPRA	PRA	PER	PER	10	11	10	10	2*	8*	6	6	8*	3*	4	4
UNK	KNW	PRA	PRA	IMPER	PER	13	12	13	12	6	6	6	6	7	6	7	6
KNW	KNW	IMPRA	PRA	IMPER	PER	15	14	15	14	8	8	8	8	7	6	7	6
UNK	KNW	PRA	IMPRA	PER	PER	12	13	12	12	4*	10*	6	8	8*	3*	6	4
UNK	KNW	PRA	PRA	PER	IMPER	19	21	20	20	6	9	8	8	13	12	12	12
KNW	UNK	IMPRA	PRA	PER	PER	13	12	13	13	5*	10*	7	9	8*	2*	6	4
KNW	UNK	PRA	PRA	IMPER	PER	21	20	21	20	14	14	14	14	7	6	7	6
UNK	UNK	PRA	PRA	PER	PER	0	19$	0	0	0	18*	0	0	0	1*	0	0
KNW	KNW	IMPRA	PRA	PER	IMPER	19$	17	16	16	6	9	8	8	13*	8	8	8
KNW	KNW	PRA	IMPRA	IMPER	PER	17	16	17	16	10	10	10	10	7	6	7	6
KNW	KNW	IMPRA	IMPRA	PER	PER	0	0	0	0	0	0	0	0	0	0	0	0
KNW	KNW	PRA	PRA	IMPER	IMPER	46	46	48	48	16*	32*	26	22	30*	14*	22	26
Skill level = 4 Av # of CPs						17	17	17	16	7	11	9	9	10	5	7	7

Notes. Each line represents a group of 4 dialogues, one for each of four different speed settings (SF,FS,SS,FF) e.g SF = John slow Mary fast). Each skill level or major subgroup measures dialogues in which agents have similar skills (e.g. KNW UNK means John is knowledgeable about the door and Mary is not). In general John always appears first. Mark $ when # of CP's > 10% deviation from the mean of the 4-group. Mark * when # of CP's > 25% deviation from the mean of the 4-group. Data sorted in descending order of skill level (award 1 point for a positive skill, 0 otherwise e.g. KNW UNK PRA PRA PER PER is 1+0+1+1+1+1 = group 5)

```
Knowledge  Action     Perception    Total CP's      CP's by John    CP's by Mary
                                  SF  FS  SS  FF  SF  FS  SS  FF  SF  FS  SS  FF

-------------------------------------------------------------------------------
KNW  UNK   PRAIMPRA   PERIMPER  13  13  12  12   7   9   8   8  6*  4   4   4
UNK  KNWIMPRA   PRAIMPER   PER  13  12  13  12   6   6   6   6   7   6   7   6
UNK  KNW   PRAIMPRA   PERIMPER  16  18  17  17   6   9   8   8  10   9   9   9
KNW  UNKIMPRA   PRA   PERIMPER  15  15  14  14   7   9   8   8   8   6   6   6
KNW  UNK   PRAIMPRAIMPER   PER  21  20  21  20  14  14  14  14   7   6   7   6
KNW  UNKIMPRAIMPRA   PER   PER   0   0   0   0   0   0   0   0   0   0   0   0
KNW  UNK    PRA   PRAIMPERIMPER  48  46  50  50  21  32  28  24  27  14*22  26
UNK  UNK   PRAIMPRA   PER   PER  23  19$21  23   7*18*  9  13  16*  1*12  10
UNK  UNK   PRA   PRA   PERIMPER   0  24$25$25$  0  17*12  12   0   7  13*13*
KNW  KNW   PRAIMPRAIMPERIMPER  48  46  50  50  18*32*26  22  30  14*24  28
KNW  KNWIMPRAIMPRA   PERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  KNWIMPRA   PRA   PERIMPER  19  21  20  20   6   9   8   8  13  12  12  12
UNK  KNW   PRAIMPRAIMPER   PER  15  14  15  14   8   8   8   8   7   6   7   6
UNK  KNWIMPRAIMPRA   PER   PER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  KNW    PRA   PRAIMPERIMPER  46  48  51  47  16*29  25  27  30*19  26  20
KNW  UNKIMPRA   PRAIMPER   PER  18  17  18  17  11  11  11  11   7   6   7   6
UNK  UNKIMPRA   PRA   PER   PER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNK   PRA   PRAIMPER   PER   0  27$  0   0   0  21*  0   0   0  6*  0   0
KNW  KNWIMPRA   PRAIMPERIMPER  46  48  52  50  16*32*26  22  30  16*26  28
KNW  KNWIMPRAIMPRAIMPER   PER   0   0   0   0   0   0   0   0   0   0   0   0

-------------------------------------------------------------------------------
Skill level = 3  Av # of CPs     26  25  27  26  11  17  14  13  15   8  13  12

-------------------------------------------------------------------------------
KNW  UNK   PRAIMPRAIMPERIMPER  48  46  50  50  21  32  28  24  27  14*22  26
KNW  UNKIMPRAIMPRA   PERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNK   PRAIMPRA   PERIMPER  31$24$27  27  11  17  14  14  20*  7*13  13  **
UNK  KNWIMPRA   PRAIMPERIMPER  46  48  51  47  16*29  25  27  30*19  26  20
UNK  KNWIMPRAIMPRAIMPER   PER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNKIMPRA   PRAIMPER   PER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  KNW   PRAIMPRAIMPERIMPER  48  50  51  49  18*31  27  29  30*19  24  20
UNK  KNWIMPRAIMPRA   PERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
KNW  UNKIMPRA   PRAIMPERIMPER  50  48  54  52  21  32  28  26  29  16*26  26
KNW  UNKIMPRAIMPRAIMPER   PER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNKIMPRA   PRA   PERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNK   PRAIMPRAIMPER   PER  28  27  28  27  13  21  17  17  15*  6*11  10
UNK  UNKIMPRAIMPRA   PER   PER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNK   PRA   PRAIMPERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
KNW  KNWIMPRAIMPRAIMPERIMPER   0   0   0   0   0   0   0   0   0   0   0   0

-------------------------------------------------------------------------------
Skill level = 2  Av # of CPs     41  40  43  42  16  27  23  22  25  13  20  19

-------------------------------------------------------------------------------
KNW  UNKIMPRAIMPRAIMPERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNK  PRAIMPRAIMPERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNKIMPRAIMPRA  PERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  KNWIMPRAIMPRAIMPERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNKIMPRA   PRAIMPERIMPER   0   0   0   0   0   0   0   0   0   0   0   0
UNK  UNKIMPRAIMPRAIMPER   PER   0   0   0   0   0   0   0   0   0   0   0   0

-------------------------------------------------------------------------------
Skill level = 1  Av # of CPs   0   0   0   0   0   0   0   0   0   0   0   0

-------------------------------------------------------------------------------
UNK  UNKIMPRAIMPRAIMPERIMPER   0   0   0   0   0   0   0   0   0   0   0   0

-------------------------------------------------------------------------------
```

Appendix 5.
Program Listing of SUPERPOWER V.5.9[26]

[26] In addition to SUPERPOWER V.5.9 reported here, the following program listings could easily be made available on request, with author's permission, in machine readable format, both on SUN POPLOG V13,13.6 running under UNIX and those marked with an asterisk on a Macintosh Plus, running under Alphapop V.1.2, containing

(1*) Cohen and Perrault(1979) .

(2*) Power (1979,1974).

(3) Houghton (1987).

(4*) Modularised version of Houghton illustrating various processes working.

(5*) Power dialogue using Houghton's language generator.

(6*) Power(1979,1974) -> SUPERPOWER version development, illustrating all the developments discussed in chapter 5.

(7*) ELIZA

(8) SHRDLU

(9*) Gazdar and Mellish (1990)

(10*) The generic procedure NEOTELL illustrating all the developments discussed in chapter 6. However, as of writing, a complete dialogue has not yet been tested.

(11*) Help files

For the time-being the attached program listing is available on a floppy disk together with the POP11 demonstration software.

```
*** run.p (1 of 1)***
;;;
;;;                    The main compilation procedure.
;;;
mycompile(popfolder<>'Power(V.5.9):VARIABLES.p');
mycompile(popfolder<>'Power(V.5.9):MACROS.p');
mycompile(popfolder<>'Power(V.5.9):SYSTEM.p');
mycompile(popfolder<>'Power(V.5.9):WORLD.p');
mycompile(popfolder<>'Power(V.5.9):CONCEPTS.p');
mycompile(popfolder<>'Power(V.5.9):AUXFUNS.p');
mycompile(popfolder<>'Power(V.5.9):EXEC.p');
mycompile(popfolder<>'Power(V.5.9):PLAN.p');
mycompile(popfolder<>'Power(V.5.9):KNOW1.p');
mycompile(popfolder<>'Power(V.5.9):KNOW2.p');
mycompile(popfolder<>'Power(V.5.9):WRITE.p');
mycompile(popfolder<>'Power(V.5.9):PLAY.p');
mycompile(popfolder<>'Power(V.5.9):PRINT.p');
mycompile(popfolder<>'Power(V.5.9):DECIPHER.p');
mycompile(popfolder<>'Power(V.5.9):ROUTINE.ACHIEVE.p');
mycompile(popfolder<>'Power(V.5.9):ROUTINE.ASSESS.p');
mycompile(popfolder<>'Power(V.5.9):ROUTINE.BASIC.p');
mycompile(popfolder<>'Power(V.5.9):ROUTINE.FINDOUT.p');
mycompile(popfolder<>'Power(V.5.9):ROUTINE.PLAN.p');
mycompile(popfolder<>'Power(V.5.9):ROUTINES.p');
mycompile(popfolder<>'Power(V.5.9):GAMES.p');
mycompile(popfolder<>'Power(V.5.9):GAME.GAME.p');
mycompile(popfolder<>'Power(V.5.9):GAME.ASK.p');
mycompile(popfolder<>'Power(V.5.9):GAME.TELL.p');
mycompile(popfolder<>'Power(V.5.9):GAME.RULE.p');
mycompile(popfolder<>'Power(V.5.9):GAME.GOAL.p');
mycompile(popfolder<>'Power(V.5.9):GAME.PLAN.p');
```

```
*** VARIABLES.p (1 of 4) ***
;;;plans
vars mpcurr,mpgoal,mptree,mpstate,mpactor,mpplan,mpcircular,mpstrategy;
vars jpcurr,jpgoal,jptree,jpstate,jpactor,jpplan,jpcircular,jpstrategy;
vars zpcurr,zpgoal,zptree,zpstate,zpactor,zpplan,zpcircular,zpstrategy;
vars spcurr,spgoal,sptree,spstate,spactor,spplan,spcircular,spstrategy;
;;;mental plans
vars zpmgoal,jpmgoal,mpmgoal,spmgoal;
vars zpgoaltype,jpgoaltype,mpgoaltype,spgoaltype;
vars z_i_have_tried,j_i_have_tried,m_i_have_tried,s_i_have_tried;
;;; Who is to start
vars zpstart;
;;; who we are,do,see and want.
vars mkme,mkyou,mkacts,mksee,mkgoal;
vars jkme,jkyou,jkacts,jksee,jkgoal;
vars zkme,zkyou,zkacts,zksee,zkgoal;
vars skme,skyou,skacts,sksee,skgoal;
vars zginformation,sginformation,jginformation,mginformation;
vars ztopics,stopics,jtopics,mtopics;
;;; whats in the world and what we believe
vars mkworld,mkrules;
vars jkworld,jkrules;
vars zkworld,zkrules;
vars skworld,skrules;

;;; what your partner does,sees,believes and wants.
vars mkxacts,mkxsee,mkxrules,mkxgoal;
vars jkxacts,jkxsee,jkxrules,jkxgoal;
vars zkxacts,zkxsee,zkxrules,zkxgoal;
vars skxacts,skxsee,skxrules,skxgoal;

;;; variables for working memory
vars mecontrol,mebox,meagain,menext,mejoint,megame1,memove1,mehold,meplace;
vars jecontrol,jebox,jeagain,jenext,jejoint,jegame1,jemove1,jehold,jeplace;
vars zecontrol,zebox,zeagain,zenext,zejoint,zegame1,zemove1,zehold,zeplace;
vars secontrol,sebox,seagain,senext,sejoint,segame1,semove1,sehold,seplace;
;;; planning index
vars macount;
vars jacount;
vars zacount;
vars sacount;
;;; actions,objects,types of objects,relative positions,
;;; objects relative positions,world structure,control game structure,
;;; control routine structure, belief structure.
vars mcrshell,munexpected_event;
vars jcrshell,junexpected_event;
vars scrshell,sunexpected_event,zunexpected_event;
vars zcacts,zctobjs,zcobjs,zctypes,zcprops,zctprops,
zcwshell,jcwshell,mcwshell,zcfshell,zcgshell,zcrshell;
;;;
;;; utterance count, when to stop, print out games, print out what type
;;; of process is going on, more printouts.
;;;
vars count,stop,pro,ppro,prw1,jt,mt,st,mspeed,jspeed,sspeed,Unexpected_events;
;;;
;;; an utterance, the state of the world.
;;;
vars wmessage,wobjects;
```

```
*** VARIABLES.p (2 of 4) ***
;;;
;;;        load Mary's dynamic variables into active memory.
;;;
define mawake();
mginformation->zginformation;mtopics->ztopics;munexpected_event->zunexpected_event;
mpgoaltype->zpgoaltype;
mpcurr->zpcurr;mpgoal->zpgoal;mptree->zptree;mpstate->zpstate;mpactor->zpactor;
mpplan->zpplan;mpcircular->zpcircular;mpstrategy->zpstrategy;macount->zacount;
mpmgoal->zpmgoal;m_i_have_tried->z_i_have_tried;
mkme->zkme;mkyou->zkyou;mkacts->zkacts;mksee->zksee;mkgoal->zkgoal;
mkworld->zkworld;mkrules->zkrules;
mkxacts->zkxacts;mkxsee->zkxsee;mkxrules->zkxrules;mkxgoal->zkxgoal;
mecontrol->zecontrol;mebox->zebox;meagain->zeagain;menext->zenext;
mejoint->zejoint;megame1->zegame1;memove1->zemove1;mehold->zehold;
meplace->zeplace;
mcrshell->zcrshell;
enddefine;
;;;
;;;        load John's dynamic variables into active memory.
;;;
define jawake();
jginformation->zginformation;jtopics->ztopics;junexpected_event->zunexpected_event;
jpgoaltype->zpgoaltype;
jpcurr->zpcurr;jpgoal->zpgoal;jptree->zptree;jpstate->zpstate;jpactor->zpactor;
jpplan->zpplan;jpcircular->zpcircular;jpstrategy->zpstrategy;jacount->zacount;
jpmgoal->zpmgoal;j_i_have_tried->z_i_have_tried;
jkme->zkme;jkyou->zkyou;jkacts->zkacts;jksee->zksee;jkgoal->zkgoal;jkworld->zkworld;
jkrules->zkrules;
jkxacts->zkxacts;jkxsee->zkxsee;jkxrules->zkxrules;jkxgoal->zkxgoal;
jecontrol->zecontrol;jebox->zebox;jeagain->zeagain;jenext->zenext;
jejoint->zejoint;jegame1->zegame1;jemove1->zemove1;jehold->zehold;
jeplace->zeplace;
jcrshell->zcrshell;
enddefine;
;;;
;;;        Swap John out.
;;;
define jasleep();
zginformation->jginformation;ztopics->jtopics;zunexpected_event->junexpected_event;
zpgoaltype->jpgoaltype;
zpcurr->jpcurr;zpgoal->jpgoal;zptree->jptree;zpstate->jpstate;zpactor->jpactor;
zpplan->jpplan;zpcircular->jpcircular;zpstrategy->jpstrategy;zacount->jacount;
zpmgoal->jpmgoal;z_i_have_tried->j_i_have_tried;
zkme->jkme;zkyou->jkyou;zkacts->jkacts;zksee->jksee;zkgoal->jkgoal;zkworld->jkworld;
zkrules->jkrules;
zkxacts->jkxacts;zkxsee->jkxsee;zkxrules->jkxrules;zkxgoal->jkxgoal;
zecontrol->jecontrol;zebox->jebox;zeagain->jeagain;zenext->jenext;
zejoint->jejoint;zegame1->jegame1;zemove1->jemove1;zehold->jehold;
zeplace->jeplace;
zcrshell->jcrshell;
enddefine;
;;;
;;;        Swap Mary out.
;;;
define masleep();
zginformation->mginformation;ztopics->mtopics;zunexpected_event->munexpected_event;zpgoaltype->mpgoaltype;
zpcurr->mpcurr;zpgoal->mpgoal;zptree->mptree;zpstate->mpstate;zpactor->mpactor;
zpplan->mpplan;zpcircular->mpcircular;zpstrategy->mpstrategy;zacount->macount;
zpmgoal->mpmgoal;z_i_have_tried->m_i_have_tried;
zkme->mkme;zkyou->mkyou;zkacts->mkacts;zksee->mksee;zkgoal->mkgoal;zkworld->mkworld;zkrules->mkrules;
zkxacts->mkxacts;zkxsee->mkxsee;zkxrules->mkxrules;zkxgoal->mkxgoal;
zecontrol->mecontrol;zebox->mebox;zeagain->meagain;zenext->menext;
zejoint->mejoint;zegame1->megame1;zemove1->memove1;zehold->mehold;
zeplace->meplace;
zcrshell->mcrshell;enddefine;
```

```
*** VARIABLES.p (3 of 4) ***
;;;
;;;         load Super Robots's dynamic variables into active memory.
;;;
define sawake();
sginformation->zginformation;stopics->ztopics;sunexpected_event->zunexpected_event;
spgoaltype->zpgoaltype;
spcurr->zpcurr;spgoal->zpgoal;sptree->zptree;spstate->zpstate;spactor->zpactor;
spplan->zpplan;spcircular->zpcircular;spstrategy->zpstrategy;sacount->zacount;
spmgoal->zpmgoal;s_i_have_tried->z_i_have_tried;
skme->zkme;skyou->zkyou;skacts->zkacts;sksee->zksee;skgoal->zkgoal;skworld->zkworld;
skrules->zkrules;
skxacts->zkxacts;skxsee->zkxsee;skxrules->zkxrules;skxgoal->zkxgoal;
secontrol->zecontrol;sebox->zebox;seagain->zeagain;senext->zenext;
sejoint->zejoint;segame1->zegame1;semove1->zemove1;sehold->zehold;
seplace->zeplace;
scrshell->zcrshell;
enddefine;
;;;
;;;         swap Super Robot out
;;;
define sasleep();
zginformation->sginformation;ztopics->stopics;zunexpected_event->sunexpected_event;
zpgoaltype->spgoaltype;
zpcurr->spcurr;zpgoal->spgoal;zptree->sptree;zpstate->spstate;zpactor->spactor;
zpplan->spplan;zpcircular->spcircular;zpstrategy->spstrategy;zacount->sacount;
zpmgoal->spmgoal;z_i_have_tried->s_i_have_tried;
zkme->skme;zkyou->skyou;zkacts->skacts;zksee->sksee;zkgoal->skgoal;zkworld->skworld;
zkrules->skrules;
zkxacts->skxacts;zkxsee->skxsee;zkxrules->skxrules;zkxgoal->skxgoal;
zecontrol->secontrol;zebox->sebox;zeagain->seagain;zenext->senext;
zejoint->sejoint;zegame1->segame1;zemove1->semove1;zehold->sehold;
zeplace->seplace;
zcrshell->scrshell;
enddefine;
;;;
;;;         odd functions that are still used before they are defined
;;;
vars zepost,zosp,zobug,zoplead;
vars mmginformation,jjginformation,mmunexpected_event,jjunexpected_event;
vars mmpgoaltype,jjpgoaltype;
vars mmpcurr,mmpgoal,mmptree,mmpstate,mmpactor,mmpplan,mmpcircular,mmpstrategy;
vars jjpcurr,jjpgoal,jjptree,jjpstate,jjpactor,jjpplan,jjpcircular,jjpstrategy;
;;;mental plans
vars jjpmgoal,mmpmgoal,mm_i_have_tried,jj_i_have_tried;
vars mmkme,mmkyou,mmkacts,mmksee,mmkgoal;
vars jjkme,jjkyou,jjkacts,jjksee,jjkgoal;
vars mmkworld,mmkrules;
vars jjkworld,jjkrules;
vars mmkxacts,mmkxsee,mmkxrules,mmkxgoal;
vars jjkxacts,jjkxsee,jjkxrules,jjkxgoal;
vars mmecontrol,mmebox,mmeagain,mmenext,mmejoint,mmegame1,mmemove1,mmehold,mmeplace;
vars jjecontrol,jjebox,jjeagain,jjenext,jjejoint,jjegame1,jjemove1,jjehold,jjeplace;
vars mmacount;
vars jjacount;
vars mmcrshell;
vars jjcrshell;
```

```
*** VARIABLES.p (4 of 4) ***
define gsave();
jginformation->jjginformation;junexpected_event->jjunexpected_event;
jpgoaltype->jjpgoaltype;
jpcurr->jjpcurr;jpgoal->jjpgoal;jptree->jjptree;jpstate->jjpstate;jpactor->jjpactor;
jpplan->jjpplan;jpcircular->jjpcircular;jpstrategy->jjpstrategy;jacount->jjacount;
jpmgoal->jjpmgoal;j_i_have_tried->jj_i_have_tried;
jkme->jjkme;jkyou->jjkyou;jkacts->jjkacts;jksee->jjksee;jkgoal->jjkgoal;jkworld->jjkworld;
jkrules->jjkrules;
jkxacts->jjkxacts;jkxsee->jjkxsee;jkxrules->jjkxrules;jkxgoal->jjkxgoal;
jecontrol->jjecontrol;jebox->jjebox;jeagain->jjeagain;jenext->jjenext;
jejoint->jjejoint;jegame1->jjegame1;jemove1->jjemove1;jehold->jjehold;
jeplace->jjeplace;
jcrshell->jjcrshell;
mginformation->mmginformation;munexpected_event->mmunexpected_event;
mpgoaltype->mmpgoaltype;
mpcurr->mmpcurr;mpgoal->mmpgoal;mptree->mmptree;mpstate->mmpstate;mpactor->mmpactor;
mpplan->mmpplan;mpcircular->mmpcircular;mpstrategy->mmpstrategy;macount->mmacount;
mpmgoal->mmpmgoal;m_i_have_tried->mm_i_have_tried;
mkme->mmkme;mkyou->mmkyou;mkacts->mmkacts;mksee->mmksee;mkgoal->mmkgoal;mkworld->mmkworld;
mkrules->mmkrules;
mkxacts->mmkxacts;mkxsee->mmkxsee;mkxrules->mmkxrules;mkxgoal->mmkxgoal;
mecontrol->mmecontrol;mebox->mmebox;meagain->mmeagain;menext->mmenext;
mejoint->mmejoint;megame1->mmegame1;memove1->mmemove1;mehold->mmehold;
meplace->mmeplace;
mcrshell->mmcrshell;
enddefine;
define gstart();
jjginformation->jginformation;jjunexpected_event->junexpected_event;
jjpgoaltype->jpgoaltype;
jjpcurr->jpcurr;jjpgoal->jpgoal;jjptree->jptree;jjpstate->jpstate;jjpactor->jpactor;
jjpplan->jpplan;jjpcircular->jpcircular;jjpstrategy->jpstrategy;jjacount->jacount;
jjpmgoal->jpmgoal;jj_i_have_tried->j_i_have_tried;
jjkme->jkme;jjkyou->jkyou;jjkacts->jkacts;jjksee->jksee;jjkgoal->jkgoal;jjkworld->jkworld;
jjkrules->jkrules;
jjkxacts->jkxacts;jjkxsee->jkxsee;jjkxrules->jkxrules;jjkxgoal->jkxgoal;
jjecontrol->jecontrol;jjebox->jebox;jjeagain->jeagain;jjenext->jenext;
jjejoint->jejoint;jjegame1->jegame1;jjemove1->jemove1;jjehold->jehold;
jjeplace->jeplace;
jjcrshell->jcrshell;
mmginformation->mginformation;mmunexpected_event->munexpected_event;
mmpgoaltype->mpgoaltype;
mmpcurr->mpcurr;mmpgoal->mpgoal;mmptree->mptree;mmpstate->mpstate;mmpactor->mpactor;
mmpplan->mpplan;mmpcircular->mpcircular;mmpstrategy->mpstrategy;mmacount->macount;
mmpmgoal->mpmgoal;mm_i_have_tried->m_i_have_tried;
mmkme->mkme;mmkyou->mkyou;mmkacts->mkacts;mmksee->mksee;mmkgoal->mkgoal;mmkworld->mkworld;
mmkrules->mkrules;
mmkxacts->mkxacts;mmkxsee->mkxsee;mmkxrules->mkxrules;mmkxgoal->mkxgoal;
mmecontrol->mecontrol;mmebox->mebox;mmeagain->meagain;mmenext->menext;
mmejoint->mejoint;mmegame1->megame1;mmemove1->memove1;mmehold->mehold;
mmeplace->meplace;
mmcrshell->mcrshell;
enddefine;
```

```
*** tests.p (1 of 1)***
;;;
;;;         How to execute SUPERPOWER along the lines described in chapter 5.
;;;
npr('Classical/Experimental/Instrumental/Unexpected 1');
define reset1;
John,John->jkme->mkyou;
Mary,Mary->mkme->jkyou;
[John in]->jkgoal; [none]->mkgoal;
[John out Mary in bolt up door shut]->wobjects;
nil->wmessage;
[ move ]->jkacts;[push slide move]->mkacts;
[[evt[robot slide]sit[any]res[undef]]
 [evt[robot push]sit[bolt up]res[door]]
 [evt[robot move]sit[door open]res[robot]]]->jcrshell;
 [[evt[robot slide]sit[any]res[undef]]
 [evt[robot push]sit[any]res[undef]]
 [evt[robot move]sit[any]res[undef]]]->mcrshell;
 [John]->jksee;
 [Mary]->mksee;
 1->count;500->stop;false->prw1;false->pro;
false->ppro;
 0->mt;
 0->jt;
;;; []->oldp;
 []->mptree;
[81 [robot slide] ]->Unexpected_events;
 []->jptree;
 1->macount;
 1->jacount;
 1->mspeed;
 3->jspeed;
 if ppro=true then [1 2 3 4 5 6 7 8 9 10 11 12 13]->ppro;else []->ppro;endif;
 if mt<mspeed and jt<jspeed then nl(1);pr('parallel processing set');
 nl(1);pr('Johns speed = ');pr(jspeed);pr('   Marys speed = ');pr(mspeed);
 else nl(1);pr('parallel processing not set');endif;
enddefine;
run1();
```

```
*** macros.p (1 of 1)***
;;;
;;;         (see Power 1974 p260)
;;;
define macro constants;
vars x;
c1:
itemread()->x;
if x/=";" then
          "vars", x, ";",
          hd(["]), x, hd(["]), "->", x, ";";
          goto c1;
endif;
enddefine;

define macro game;
vars name,x,g,l;
itemread()->name;
nil,nil->g->l;
erase(itemread());
g1:
itemread()->x;
if x="b" then "black"->x endif; if x="w" then "white"->x endif;
if x/="end"
then if x/=";"
          then if x/="."
                    then l<>[%x%]->l
                    endif;
                    goto g1;
          else g<>[%hd(l),tl(l)%]->g;nil->l; goto g1;
          endif
else  "vars",name,";",g,"->",name;
endif;
enddefine;

define macro routine();
"game";
enddefine;
```

```
*** SYSTEM.p (1 of 3)***
;;;
;;;        See (Power 1974 p261)
;;;
constants John Mary Dick both door bolt in out open shut
      up down name kind colour place entries mark white
      black achieved failed notyet push slide move sit res
      evt expectreply goaltype actor;
;;;        memb is true if x is member of l and false if not;
;;;        e.g 1. if memb("a",[a b c]) then npr([true]) else npr([false]) endif;
;;;        e.g 2. memb("d",[a b c])=>;
;;;
define memb(x,l);
         while (ispair(l) and hd(l)/=x) do; tl(l)->l; endwhile;
         ispair(l);
enddefine;
;;;
;;;        runs a piece of program in list l
;;;        popval is the same as eval now
;;;
vars eval;  popval->eval;
;;;
;;;         returns item succeeding x in l. only works for odd words.
;;;         e.g. npr(suc([a b c d e f g h ],"e"));
;;;
define suc( l,x);
while (ispair(l) and hd(l)/=x) do; tl(tl(l))->l; endwhile;
if null(l) then undef else hd(tl(l)); endif;
enddefine;
;;;
;;;        returns item one succeeding  of x in l. Only works
;;;        for triplets.
;;;
define suc2( l,x);
while (ispair(l) and hd(l)/=x) do; tl(tl(tl(l)))->l; endwhile;
if null(l) then undef else hd(tl(l)); endif;
enddefine;
;;;
;;;        returns item  succeeding the succedant of x in l. Only works
;;;        for triplets.
;;;
define suc3( l,x);
while (ispair(l) and hd(l)/=x) do; tl(tl(tl(l)))->l; endwhile;
if null(l) then undef else hd(tl(tl(l))); endif;
enddefine;
;;;
;;;        returns item preceding n in l. works for odd words (change tl(tl
;;;         to tl for odd and even words.
;;;        e.g. npr(pre([a b c d e f g h ],"e"));
;;;
;;;
define pre(l,n);
while ispair(l) and hd(tl(l))/=n do; tl(tl(l))->l; endwhile;
if null(l) then undef else hd(l); endif;
enddefine;
;;;
;;;
;;;        deletes n and its successor from l
;;;        e.g. 1. npr(del([[a b] [c d] [e f] [g h] [i j]],[c d]));
;;;         e.g. 2. npr(del( [a b c d e f g h],"c"));
;;;
define del(l,n);
 if not(memb(n,l)) then return(l);endif;
 while hd(l)/=n do; tl(tl(l)) <> [%hd(l),hd(tl(l))%]->l;endwhile;
 tl(tl(l));
enddefine;
```

```
*** SYSTEM.p (2 of 3)***
;;;
;;;         deletes n and its two successors from l
;;;
define del2(l,n);
 if not(memb(n,l)) then return(l);endif;
  while hd(l)/=n do; tl(tl(tl(l))) <> [%hd(l),hd(tl(l)),hd(tl(tl(l)))%]->l;endwhile;
 tl(tl(tl(l)));
enddefine;


;;;
;;;
;;;
;;;         replaces successor of n with x in l
;;;         e.g. npr(rep([[a b] [c d] [e f] [g h] [i j]],[e f],[nnnnn]));
;;;
;;;
define rep(l,n,x);
[%n,x%]<>del(l,n);
enddefine;
;;;
;;;         replaces successor of n with x,y in l
;;;
define rep2(l,n,x,y);
[%n,x,y%]<>del2(l,n);
enddefine;


;;;
;;;
;;;         substitutes x after n in list named w
;;;         e.g. [[a b] [c d] [e f] [g h]]->g;
;;;         sub("g",[e f], [wow wof]);
;;;       g=>;

;;;
;;;
define sub(w,n,x);
rep(valof(w),n,x)->valof(w);
enddefine;
;;;
;;;          substitutes x,y after n in list named w
;;;
define sub2(w,n,x,y);
rep2(valof(w),n,x,y)->valof(w);
enddefine;


;;;
;;;
;;;         exchanges x2 for x1 at all levels of list l
;;;         e.g. xch([a b [ c b ] d],"b","z");=>;
;;;
   define xch (l,x1,x2) -> ll;
             nil->ll;rev(l)->l;
   while ispair(l) do;
   if islist(hd(l)) then xch(hd(l),x1,x2);
   elseif hd(l)=x1 then x2 else hd(l)
   endif; ::ll->ll;tl(l)->l; endwhile;
   enddefine;
;;;
;;;         print routines
;;;
define prs(x);  spr(x); enddefine;

define prl(x);
while ispair(x) do;pr(hd(x));tl(x)->x; endwhile;
enddefine;
```

*** SYSTEM.p (3 of 3)***

```
define pra(l);
;;;nl(1);
while ispair(l) do; sp(2); pr(hd(l));sp(1);pr(hd(tl(l)));tl(tl(l))->l;
endwhile;enddefine;

define prb (l);
vars x y;
valof(suc(l,name))->x;
suc(l,entries)->l;
rev(l)->l;
prs('entries');
          while ispair(l) do
                    if ispair(hd(l)) then
                              nl(1);sp(1);pr(hd(tl(l)));prs('. ');
                              suc(x,hd(tl(x)))->y;
                              if ispair(y) then
                                        tl(y)->y;
          if memb(hd(y),[white black])
          then hd(tl(y)) else hd(y);
          endif;
       else ' - '
       endif; .pr;
       sp(2); pr(hd(l));
    endif;
    tl(tl(l))->l;
  endwhile; nl(1);
enddefine;

define prg (l);
nl(1);prs('game ');pr(suc(l,name));
nl(1);prs('current place:');pr(suc(l,place));
nl(1);prs('my colour: ');pr(suc(l,colour));
nl(1);prb(l);
enddefine;

define prf (l);
vars x;
nl(1);prs('routine ');pr(suc(l,name));
nl(1);prs('current place:');pr(suc(l,place));
nl(1);prs('marked place: ');
if null(zeplace) then sp(1);"none"
elseif
hd(zeplace)=suc(l,name) then hd(tl(zeplace)) else sp(1);"none";endif;
pr();nl(1);prb(l);
enddefine;

define prc(l);
 if suc(l,kind)="game" then prg(l) else prf(l) endif;
 enddefine;

define prw;
true->prw1;prs('(State of the world is now ');pr(wobjects);prs(')');
if pro  then nl(1) endif;nl(1);
enddefine;
```

```
*** WORLD.p (1 of 1)***
;;;
;;;
;;;        *****   Objects in the world     ********
;;;      see  (Power 1974 p264)
;;;
;;;
;;; The functions wmove,wpush and wslide defined below
;;;  represent the actual laws of the universe: that is the
;;;  the three kinds of actions and their consequences
;;;
;;;
define wpos x;
suc(wobjects,x);
enddefine;

define wsub n x;
sub("wobjects",n,x);
enddefine;

define wmove (robot);
if wpos(door)=open
then if wpos(robot)=in or wpos(robot)="in"
   then wsub(robot,out)
   else wsub(robot,"in")
   endif;
endif;
enddefine;

define wpush (robot);
if wpos(bolt)=up
then if wpos(door)=open
   then wsub(door,shut)
   else wsub(door,open)
   endif;
endif;
enddefine;

define wslide (robot);
if wpos(robot)=in or wpos(robot)="in" or wpos(robot)=undef
then if wpos(bolt)=up
   then wsub(bolt,down)
   else wsub(bolt,up)
   endif;
endif;
enddefine;
```

```
*** CONCEPTS.p (1 of 1)***
;;;
;;;
;;;     ****    [concepts]  (Power 1974 p 265)
;;;
[move slide push]                                 ->zcacts;
[John Mary door bolt]                              ->zcobjs;
[slide [bolt] move [robot] push [door]]                    ->zctobjs;
[robot door bolt]                               ->zctypes;
[in out up down open shut]                           ->zcprops;
[robot[in out] door[open shut] bolt[up down]]                ->zctprops;
[John undef undef Mary undef undef bolt undef undef door undef undef]
->zcwshell;
[kind routine name undef place undef entries undef]          ->zcfshell;
[kind game name undef place undef entries undef colour undef]->zcgshell;


 define zctypof x;
 if x=undef then return(x) endif;
 if memb(x,zctypes) then x else "robot" endif;
 enddefine;

define zcobject x;
if islist(x) and length(x)>1 then return(false) endif;
if ispair(x) then hd(x)->x;endif;
memb(x,[nothing]<>[robot]<>zcobjs);
enddefine;

 define zcpropof(p,x);
 memb(p,suc(zctprops,x));
 enddefine;


 define zcevent(x);
 if islist(x) and length(x)>1 and memb(hd(tl(x)),zcacts)
 then true else false endif;
 enddefine;

define zctgame;
vars l,c;
zecontrol->l;0->c;
if suc(hd(l),kind)="routine" and zehold/=[] then return(0);endif;
while ispair(l) do
if suc(hd(l),name)="zrassess" or suc(hd(l),name)="zrachieve"
 then return(c);endif;
if suc(hd(l),kind)="game" and suc(hd(l),expectreply)=true  then
0->l;1+c->c;endif;
if c=0 and suc(hd(l),kind)="game" and suc(hd(l),expectreply)=false then
 return(c);endif;tl(l)->l;
endwhile;c
enddefine;

define zcgame(x);
 memb(x,[zgplan zgtest zgexperiment zggoal zggame zgrule zgassess zgtell zgcheck zgask]);
 enddefine;
```

```
*** KNOW1.p (1 of 3)***
;;;
;;;         The first batch of functions are used to initialise the
;;;          above variables;
;;;
;;;          This function takes a fresh look at what one can see
;;;
define zklook;
vars v;
zksee->v;
while ispair(v) do sub("zkworld",hd(v),wpos(hd(v)));tl(v)->v;endwhile;
enddefine;
;;;
;;;         This function demotes the status of what you can see to undefined
;;;         if you have either estimated or being told about the position of
;;;         an object.
;;;
define demote;
vars v n s;
zkworld->v;
while ispair(v) do
if hd(tl(tl(v)))/="seen" then
 if hd(tl(tl(v)))="told" or hd(tl(tl(v)))="inferred" then
   sub2("zkworld",hd(v),hd(tl(v)),"estimated");
 endif;
;;;  if hd(tl(tl(v)))="estimated" then
;;;     sub2("zkworld",hd(v),hd(tl(v)),"undef");
;;;  endif;
endif;
tl(tl(tl(v)))->v;
endwhile;
enddefine;
;;;
;;;          The new version of zklook that checks that preconditions are true
;;;          before one can see an object.
;;;
define zklook;
vars v;
zksee->v;
demote();
while ispair(v) do
if ispair(tl(v)) and islist(hd(tl(v))) then
    if wpos(hd(hd(tl(v))))=hd(tl(hd(tl(v)))) then
    sub2("zkworld",hd(v),wpos(hd(v)),"seen"); ;;; precondition true
    else
    sub2("zkworld",hd(v),undef,"seen");        ;;; precondition untrue
    endif;
    tl(v)->v;
else
    sub2("zkworld",hd(v),wpos(hd(v)),"seen"); ;;; no precondition at all
endif;
tl(v)->v;
endwhile;
enddefine;
;;;
;;;  This function prepares the k variables at start of run
;;;
define zkprep;
zcwshell->zkworld;zklook();
zcrshell->zkrules;[]->zehold;[]->ztopics;
[John undef Mary undef bolt undef door undef]->zkxsee;
false->z_i_have_tried;[]->zginformation;
nil->zkxgoal;
if zpstart/="Super" then nil->zkxacts;
nil->zkxrules;endif;
enddefine;
```

```
*** KNOW1.p (2 of 3)***
;;;
;;;
;;;          The next functions are concerned with finding out whether goals ]
;;;          are done or or whether they can be done; sometimes a function sets up
;;;          game zgask in order to get the answer.
;;;
;;;
;;;          This function finds the truth value of statement s in world w
;;;
define zktval(s,w);
vars p;
unless ispair(s) then return(undef); endunless; ;;; defend against bad arg.  swd
if suc3(w,hd(s))="seen" or suc3(w,hd(s))="told" or suc3(w,hd(s))="inferred"
   then
   suc2(w,hd(s))->p; else undef->p;
 endif;
if p=undef then undef else p=hd(tl(s)) endif;
enddefine;
;;;
;;;
;;;          This function finds out whether s is true or is the case
;;;          in own resources.
;;;
;;;
define zkis1(s);
zktval(s,zkworld);
enddefine;
;;;
;;;          Finds out if goal g can be done using own resources
;;;          should return true or false.  but suc does not.
;;;
define zkcan1(g);
if zcevent(g)
     then
                              if hd(g)=zkme
                              then memb(hd(tl(g)),zkacts)
                              else suc(zkxacts,hd(tl(g)))
                              endif;
           else     if suc(zpstate,pre(zpgoal,g))=[failed] then false else true
                              endif;
           endif;
enddefine;
;;;
;;;          Finds out if s is the case, asking if necessary unless it is
;;;          known that partner also doesn't know.
;;;
define zkis(s);
vars a;
zkis1(s)->a;
if a/=undef then return(a) endif;
[%(zpcurr+1)::s%]<>zpmgoal->zpmgoal;
s->zemove1;
zeload("zrfindout"); [asked];
enddefine;
;;;
;;;          Finds out if goal g can be done, asking if necessary
;;;
define zkcan(g);
vars a;
zkcan1(g)->a;
if a/=undef then a else
[%(zpcurr+1)::g%]<>zpmgoal->zpmgoal;
zeplay("zgask",[can]<>g); [asked] endif;
enddefine;
```

```
*** KNOW1.p (3 of 3)***
;;;
;;;
;;;          Finds out and enters state of goal g, asking any necessary questions
;;;
;;;
define zkstate(g);
vars a;
zkis(g)->a;
if a=[asked] then return
elseif a=undef then zaenter([failed]); return
elseif a=1 or a=true then zaenter([achieved]); return;
endif;
zkcan(g)->a;
if a=[asked] then return;
elseif a=0 or a=false then zaenter([failed])
else zaenter([notyet])
endif;
enddefine;
;;;
;;;          Checks to see whether something has already been asked,
;;;          or just been told.
;;;
define zkinfoasked(i);
vars z t a;
zginformation->z;
while ispair(z) do
 hd(z)->t;
 if hd(t)="zgask" or hd(t)="zgtell" then
    hd(tl(tl(t)))->a;hd(tl(t))->t;
    if tl(t)=[] or tl(i)=[] then
      if t=i then return([asked]) endif;
    else
      if tl(tl(t))=i or (tl(tl(t))=zkopp(i) and a=[no]) or
         (tl(t)=zkopp(i) and a=[no]) or tl(t)=i or t=i
        then return([asked]);
      endif;
    endif;
   elseif hd(t)="zgrule" then
    hd(tl(t))->t;
    if t=i then
     return([asked]);
    endif;
 endif;
tl(z)->z;
endwhile;
return([notasked]);
enddefine;
```

```
*** KNOW2.p (1 of 5) ***
;;;
;;;         The third batch of functions is for accessing and
;;;         updating the theory of how the world works, and the model
;;;         of the other robots theory. these theories are held in zkrules
;;;         and zkxrules respectively.(Power 1974 p269)
;;;


;;;
;;;         Given list of rules l and event e, returns relevant rule, or undef
;;;         if there is none.
;;;
define zkrule1(l,e);
while ispair(l) and suc(hd(l),evt)/=([robot]<>tl(e)) do tl(l)->l; endwhile;
if null(l) then undef else hd(l) endif;
enddefine;
;;;
;;;         Given rule r and event e, makes rule specific to e
;;;
define zkspec(r,e);
xch(r,"robot",hd(e));
enddefine;
;;;
;;;         Finds own rule for event e
;;;
define zkrule(e);
zkrule1(zkrules,e);
enddefine;
;;;
;;;         Finds own specific rule for event e
;;;
define zksrule(e);
zkspec(zkrule(e),e);
enddefine;
;;;
;;;         Finds partners rule for e, or undef if not known.
;;;
define zkxrule(e);
zkrule1(zkxrules,e);
enddefine;
;;;
;;;         Finds partners specific rule for e, if known, and undef if not.
;;;
define zkxsrule(e)->r;
zkxrule(e)->r;
if r/=undef then zkspec(r,e)->r endif;
enddefine;
;;;
;;;         Deletes rule for event e from list l (Power 1974 p270)
;;;

define zkdel(l,e)->ll;
nil->ll;
while ispair(l)
do if suc(hd(l),evt)/=([robot]<>tl(e))
then ll<>[%hd(l)%]->ll
endif;
tl(l)->l; endwhile;
enddefine;
```

```
*** KNOW2.p (2 of 5)***
;;;
;;;         Gives inverse form of rule.
;;;
define zkinvert(r);
vars e,s,c;
suc(r,evt)->e; suc(r,sit)->s; suc(r,res)->c;
if s=[any] then return(r) endif;
if c=[nothing] then suc([move robot slide bolt push door],hd(tl(e))) :: nil
else [nothing]
endif;->c;
suc(zctprops,hd(s))->r;
if hd(tl(s))=hd(r) then hd(tl(r)) else hd(r) endif; ->r;
[%evt,e,sit,[%hd(s),r%],res,c%];
enddefine;


;;;
;;;         Puts new rule r in list named n.
;;;
define zkadd1(n,r);
if suc(r,sit)/=[any] and suc(r,res)=[nothing]
then zkinvert(r)->r endif;
r::(zkdel(valof(n),suc(r,evt)))->valof(n);
enddefine;
;;;
;;;          Puts rule r in robots theory.
;;;
define zkadd(r);
pr(zkme);pr(' adding rule in (zkrules)   : ');npr(r);
zkadd1("zkrules",r);
enddefine;
;;;
;;;         Puts rule r in model of partners theory.(Power 1974 p271)
;;;
define zkxadd(r);
pr(zkme);pr(' adding rule in (zkxrules)   : ');npr(r);
zkadd1("zkxrules",r);
enddefine;
;;;
;;;          Predicted result of event in world w by specific rule r.
;;;
define zkpred1(w,r);
vars t,s;
suc(r,sit)->s;
if s=[any] then return(suc(r,res)) endif;
zktval(s,w)->t;
if t=undef then [undef]<>s
elseif t or t=1 then suc(r,res) else [nothing]
endif;
enddefine;
;;;
;;;          Result of event e in world w as predicted by own rules.
;;;
define zkpred(e,w);
zkpred1(w,zksrule(e));
enddefine;
;;;
;;;          Result of event e in world w as predicted by partners rules if
;;;          known: undef if not. really, of course, the result is that
;;;          predicted by model of partners theory.
;;;
define zkxpred(e,w);
vars r;zkxsrule(e)->r;
if r=undef then undef else zkpred1(w,r) endif;
enddefine;
```

```
*** KNOW2.p (3 of 5)***
;;;
;;;
;;;         Returns false if all actions in rules produce no result and true otherwise
;;;          (Power 1974 p271)
;;;
define zkfluid;
vars r;  zkrules->r;
while ispair(r)  and suc(hd(r),res)=[nothing] do tl(r)->r endwhile;
ispair(r);
enddefine;
;;;
;;;         Finds opposite of situation s.
;;;
define zkopp(s);
vars p;
if zcevent(s) or zcobject(s) then return(s);endif;
suc(zctprops,zctypof(hd(s)))->p;
if hd(tl(s))=hd(p) then hd(tl(p)) else hd(p) endif;::[%hd(s)%];rev();
enddefine;
;;;
;;;
;;;         Finds rule to achieve specified result c, returning undef
;;;         if none exists (p272). zkinvert added by CGB 'cos you
;;;        won't find a rule if the inverse res is nothing.
;;;
define zkres (c);
vars t;
zkrules->t;
while ispair(t) and suc(hd(t),res)/=c and suc(zkinvert(hd(t)),res)/=c then tl(t)->t endwhile;
if null(t) then undef else
if suc(hd(t),res)=[nothing] then zkinvert(hd(t)) else hd(t) endif; endif;
enddefine;
;;;
;;;         Finds rule in which the situation is true
;;;
define zksit (c);
vars t;
zkrules->t;
while ispair(t) and suc(hd(t),sit)/=c  then tl(t)->t endwhile;
if null(t) then undef else  hd(t) endif;
enddefine;
;;;
;;;         Finds rule in which the event is true
;;;
define zkevt (c);
vars t;
zkrules->t;
while ispair(t) and suc(hd(t),evt)/=c  then tl(t)->t endwhile;
if null(t) then undef else  hd(t) endif;
enddefine;
```

*** KNOW2.p (4 of 5)***

```
;;;
;;;         Changes theory to fit experience. e is the event which has just
;;;         occurred and c is its consequence (or res). ee, cc, ss are the
;;;          evt, res and sit of the old rule rr. the function alters rr and
;;;          puts the new rule in zkrules.
;;;
define zkponder(e,c);
vars ee,cc,ss,rr,l,x;
zkrule(e)->rr;
if c/=[nothing] then
     [%zctypof(hd(c))%]->c;          ;;; converts Mary to robot in new rule
endif;
suc(rr,evt)->ee;suc(rr,sit)->ss;suc(rr,res)->cc;
if cc=[undef] then zkadd(rep(rr,res,c));return(1) endif;
if cc/=c and c/=[nothing] and cc/=[nothing]
   then rep(rr,res,[%zctypof(hd(c))%])->rr;
   zkadd(rep(rr,sit,[any]));
   return(1);
   endif;
c->cc;rep(rr,res,cc)->rr;
zctypes->l;
while hd(l)=hd(cc) or hd(l)=hd(ss) do tl(l)->l endwhile;
if hd(l)="robot" then hd(e) else hd(l) endif;->e;
if suc2(zkworld,e)=undef then zeplay("zgask",zamakeq(e)); return([asked]);
endif;
[%hd(l),suc2(zkworld,e)%]->ss;
zkadd(rep(rr,sit,ss));
1;
enddefine;
;;;
;;;         Judges whether plan p will achieve goal g.( Power 1974 p273)
;;;
define zkjudge(p,g);
vars e,s,w,k;
if not(.zkfluid) then return(undef) endif;
zpevt(p)->e;zpsit(p)->s;
zkworld->w;
if s/=undef then rep2(w,hd(s),hd(tl(s)),"told")->w endif;
zkpred(e,w)->s;
if hd(s)=undef and tl(s)/=[]
 and suc(zpgoaltype,zpnextgl())/="test"
then  if zkinfoasked(tl(s))/=[asked] then zkis(tl(s));endif;
;;;zeplay("zgask",[is]<>tl(s));[asked]
else hd(s)=hd(g)
endif;
if suc(zpgoaltype,zpnextgl())="test" then
suc(zkevt(zctypof(hd(zpevt(p)))::tl(hd(tl(p)))),res)->k;
if k=[nothing] or k=[undef] then else
if k=[robot] then [%hd(p)%]->k;endif;
if zkinfoasked(tl(zamakeq(hd(k))))/=[asked] then tl(zamakeq(hd(k))).zkis;endif;endif;
endif;
enddefine;
```

```
*** KNOW2.p (5 of 5) ***

;;;
;;;          A refinement on zktval to decide whether situation s is true in world w
;;;
define zkdiff(s,w);
if s=[any] then return(true);endif;
;;;
;;;          This one doesn't give strict truth but at least one of the robots
;;;          fit the bill.
;;;
if hd(s)="robot" then zkme::tl(s)->s;if zktval(s,w)=true then return(true);endif;
zkyou::tl(s)->s;return(zktval(s,w));endif;
zktval(s,w);
enddefine;

define zkbetter(rule1,rule2);
vars s1,r1,s2,r2,p1,p2;
if rule1=undef then return(false) elseif rule2=undef then return(true) endif;
if suc(rule1,res)=[undef] then return(false) elseif
suc(rule2,res)=[undef] then return(true) endif;
if zaequal(rule1,rule2) then return(undef) endif;
if suc(rule1,res)=[nothing] then zkinvert(rule1)->rule1;endif;
if suc(rule2,res)=[nothing] then zkinvert(rule2)->rule2;endif;
suc(rule1,res)->r1;suc(rule2,res)->r2;suc(rule1,sit)->s1;suc(rule2,sit)->s2;
if zkdiff(s1,zkworld) then
 if r1=[nothing] then
  if s1=[any]  then
  4->p1;else 2->p1;
  endif;
 else
  if s1=[any]  then
  3->p1 else 2->p1;
  endif;
 endif;
else
1->p1;
endif;
if zkdiff(s2,zkworld) then
 if r2=[nothing] then
  if s2=[any]  then
  4->p2;else 2->p2;
  endif;
 else
  if s2=[any]  then
  3->p2 else 2->p2;
  endif;
 endif;
else
1->p2;
endif;
pr(p1);pr(' rule 1 :');npr(rule1);
pr(p2);pr(' rule 2 :');npr(rule2);
if p1=p2 then undef else p1<p2 endif;
enddefine;
```

*** PLAN.p (1 of 4) ***

```
;;;
;;;          (Power 1974 274-275) These are general routines for tidying up a joint plan tree.
;;;          and are largely called at the end of routine achieve.
;;;
;;;
;;;          Read plan tree from the last goal back to the first goal.
;;;
define zpreadtree();
vars c;
zpnextgl()->c;
while c/=0 do
pr(c);pr('  ');npr(suc(zpgoal,c));
zpprevious(c)->c;
endwhile;
enddefine;
;;;
;;;          Returns index number of next goal to attempt.
;;;
define zpnextgl->n1;
vars n2;[0]->n2;
while n2/=undef do  hd(n2)->n1; suc(zptree,n1)->n2 endwhile;
enddefine;
;;;
;;;          Returns parent of node n in zptree.
;;;
define zpparent (n);
vars t; zptree->t;
while ispair(t) and not(memb(n,hd(tl(t)))) do tl(tl(t))->t endwhile;
if null(t) then undef else hd(t) endif;
enddefine;
;;;
;;;          Returns index number of previous goal.
;;;
define zpprevious (n);
vars t;zptree->t;
while ispair(t) and not(memb(n,hd(tl(t)))) do tl(tl(t))->t endwhile;
if null(t) then undef else if ispair(tl(hd(tl(t)))) and not(memb(n,tl(hd(tl(t)))))
 then
hd(tl(hd(tl(t)))) else hd(t) endif;endif;
enddefine;
;;;
;;;          Returns a list of plans and goals after p.In order of seniority. i.e. the
;;;          plan at the back of the list is the one highest up the tree and the goal
;;;          nearest to the main goal.
;;;
define zplist(p);
vars t l;
zptree->t;suc(zpgoaltype,p)::nil->l;suc(zpgoal,p)::l->l;
while hd(hd(tl(t)))/=p
do
suc(zpgoaltype,hd(hd(tl(t))))::l->l;
suc(zpgoal,hd(hd(tl(t))))::l->l;
if ispair(tl(hd(tl(t)))) then
if hd(tl(hd(tl(t))))/=p then
suc(zpgoaltype,hd(tl(hd(tl(t))))):l->l;
suc(zpgoal,hd(tl(hd(tl(t))))):l->l;
else
return(l);
endif;
endif;
tl(tl(t))->t;
endwhile;
return(l);
enddefine;
```

```
*** PLAN.p (2 of 4) ***
;;;
;;;         Takes a list of Conversational games c and a plan tree t and finds
;;;          the first occurrence in the plan tree.
;;;
define zpworkthrupublic(t,c);
vars h i s z gt;t->s;10000->h;10000->i;
while ispair(c) do
hd(tl(tl(tl(hd(c)))))->gt;
if hd(hd(c))="zgask" and hd(hd(tl(hd(c))))/="can" then
if hd(tl(tl(hd(c))))=[no] then
tl(hd(tl(hd(c)))).zkopp->z;
zpsearch(s,z,gt)->h;endif;
if hd(tl(tl(hd(c))))=[yes] then
tl(hd(tl(hd(c))))->z;
zpsearch(s,z,gt)->h;endif;
endif;
if hd(hd(c))="zgtell" then
if ((hd(hd(tl(hd(c))))=false or hd(hd(tl(hd(c))))=0) and
(hd(tl(tl(hd(c))))=[yes] or hd(tl(tl(hd(c))))=[undef])) or ((hd(hd(tl(hd(c))))=true or hd(hd(tl(hd(c))))=1) and
hd(tl(tl(hd(c))))=[no])
then
tl(tl(hd(tl(hd(c))))).zkopp->z;
zpsearch(s,z,gt)->h;endif;
if ((hd(hd(tl(hd(c))))=true or hd(hd(tl(hd(c))))=1) and
 (hd(tl(tl(hd(c))))=[yes] or hd(tl(tl(hd(c))))=[undef]))
or ((hd(hd(tl(hd(c))))=false or hd(hd(tl(hd(c))))=0) and hd(tl(tl(hd(c))))=[no])
then
tl(tl(hd(tl(hd(c)))))->z;
zpsearch(s,z,gt)->h;endif;
if zcevent(hd(tl(hd(c)))) then
hd(tl(hd(c)))->z;
zpsearch(s,z,gt)->h;endif;
endif;
npr(z);
if h<i then h else i endif;->i;
tl(c)->c;
endwhile;
return(i);
enddefine;
;;;
;;;         This function works through zkworld ones private view of the world
;;;         and zpgoal the plan tree and decides the earliest plan that has
;;;          succeeded.
;;;
define zpworkthruprivate(t,c);
vars h i s z;t->s;10000->h;10000->i;
while ispair(c) do
[^(hd(c)) ^(hd(tl(c)))]->z;
if memb(hd(tl(tl(c))),[seen told inferred estimated]) then
  zpsearch(s,z,[any])->h;
else
  10000->h;
endif;
if h<i then h else i endif;->i;
tl(tl(tl(c)))->c;
endwhile;
return(i);
enddefine;
```

```
*** PLAN.p (3 of 4) ***
;;;
;;;         Searches through plan tree to find first occurence of goal g.
;;;
define zpsearch(t,g,gt);
vars c q;t->q;10000->c;
while ispair(q) do
if g=hd(tl(q)) or
 (suc(zpgoaltype,hd(q))="test" and  islist(hd(tl(q))) and g=hd(tl(q)).zkopp)
            then if zcevent(g) then
                 if suc(zpgoaltype,hd(q))=hd(gt) or gt=[any] then
                 if c>hd(q) then hd(q)->c;endif;
                 endif;
               else
                 if c>hd(q) then hd(q)->c;endif;
               endif;
endif;
tl(tl(q))->q;
endwhile;
return(c);
;;;if c/=10000 then return(c) else return(zpnextgl());endif;
enddefine;
;;;
;;;         Hangs plan p on node n of zptree with goaltype t.
;;;
define zphang(p,n,t);
vars l,q;
nil->l;
if suc(zpgoaltype,zpnextgl())=t
and memb(suc(zpgoal,zpnextgl()),p) then return; endif;
while ispair(p) do
zaindex()->q;l<>[%q%]->l;
sub("zpactor",q,hd(p));tl(p)->p;
sub("zpgoal",q,hd(p));tl(p)->p;
sub("zpstate",q,notyet);
sub("zpgoaltype",q,t);
endwhile;
sub("zptree",n,l);
enddefine;
;;;
;;;         Deletes all tree below node n.
;;;
define zpdelete(n);
vars s;
zptree,n.suc->s;
if s=undef then return else del(zptree,n)->zptree endif;
while ispair(s) do zpdelete(hd(s));tl(s)->s endwhile;
enddefine;

;;;
;;;         Removes terminal node n from tree.
;;;
define zpchop (n);
vars p,s;zpparent(n)->p;zptree,p.suc.tl->s;
if null(s) then zpdelete(p) else sub("zptree",p,s) endif;
enddefine;
;;;
;;;         Returns node of plan in zpgoal.
;;;
define zpfplan (p);
vars t; zpgoal->t;
while ispair(t) and not(p = hd(tl(t))) do
 tl(tl(t))->t endwhile;
if null(t) then undef else hd(t) endif;
enddefine;
```

```
*** PLAN.p (4 of 4)
;;;
;;;          Brings about event e.
;;;
define zpdo (e);
hd(e);hd(tl(e))->e;
if e=push then wpush()
elseif e=slide then wslide()
elseif e=move then wmove()
endif;
nl(1);prw();nl(1);
enddefine;
;;;
;;;          Finds the event in plan p.
;;;
define zpevt (p);
hd(rev(p));
enddefine;
;;;
;;;          Finds the sitn, if any, in plan p.
;;;
define zpsit (p);
if length(p)=2 then undef else hd(tl(p)) endif;
;;;if length(p)<2 then undef else hd(tl(p)) endif;
enddefine;

;;;
;;;          Clears away information about old goals.
;;;

define zpprune;
vars t,l,p,q;
nil->l;zptree->t;
while ispair(t) do l<>hd(tl(t))->l; tl(tl(t))->t endwhile;
[zpgoal zpactor zpstate zpgoaltype]->p;
        until null(p) do
                valof(hd(p))->t;nil->q;
                while ispair(t) do
                        if memb(hd(t),l) then [%hd(t),hd(tl(t))%]<>q->q endif;
                        tl(tl(t))->t
                endwhile;
                q->valof(hd(p));
                tl(p)->p;
        enduntil;
enddefine;
```

```
*** ROUTINES.p (1 of 1) ***
;;;
;;;          (Power 1974, p276) ROUTINES
;;;
routine zrbasic;
1.  * zrmaingl;
2.  ^ zrprep;
3.  ^ zrprep1;
4.  * zrachgl;
5.  ^ zrmove1;
6.  * zrappeal;
7.  ^ zrprep2;
8.  * zrachgl;
9.  ^ zrhalt;
end;

routine zrachieve;
1.  * zrcurrgl;
2.  * zractor;
3.  * zrstate;
4.  * zrkind;
5.  * zrplan1;
6.  ^ zrreturn;
7.  * zrprune;
8.  ^ zrclean;
end;

routine zrassess;
1. * zrgoalintended;
2. * zraction;
3. * zractor;
4. * zrprecondition;
5. * zrobject;
6. * zrrules;
7. * zrbefore;
8. * zract;
9. * zrresult;
10. * zrparent;
11. * zrlesson;
12. * zrprune;
13. ^ zrclean;
end;

routine zrplan;
 1. * zrgoal;
 2. * zrrule;
 3. * zrspec;
 4. * zrsit;
 5. ^ zrcomp;
end;

routine zrfindout;
1. * zrinformation;
2. * zrask;
3. * zrtry;
4. ^ zrconclude;
end;
```

```
*** ROUTINE.BASIC (1 of 1) ***
;;;
;;;         1. Enters the main goal.
;;;
define zrmaingl;
zaenter(zkgoal);
enddefine;
;;;
;;;         2. Prepares world model initialises k variables.
;;;
define zrprep;
.zkprep;zagoto(3);
enddefine;
;;;
;;;         3. If there is a main goal, prepares for solo attempt.
;;;
define zrprep1;
vars n;
if zaentry(1)=[none] then return(zagoto(9)) endif;
.zaindex->n;
nil,nil,nil,nil,nil,nil->zpgoaltype->zpstate->zpgoal->zptree->zpactor->zpmgoal;
sub("zpmgoal",n,zkgoal);sub("zpgoal",n,zkgoal);sub("zptree",0,[%n%]);
sub("zpstate",n,notyet);sub("zpactor",n,zkme);sub("zpgoaltype",n,"plan");
false->zejoint; zagoto(4);
enddefine;
;;;
;;;         4 & 8. Attempts goal, entering [achieved] or [failed].
;;;
define zrachgl;
zeload("zrachieve");
enddefine;
;;;
;;;         5. If solo attempt succeeds, halt; if not, seek help.
;;;
define zrmove1;
if zaentry(4)=[achieved]  then zagoto(9) else zagoto(6) endif;
enddefine;
;;;
;;;         6. Arranges game to appeal for help, game enters yes or no.
;;;
define zrappeal;
zeplay("zggoal",zkgoal);
enddefine;
;;;
;;;         7. If appeal succeeds, prepare for joint attempt. if not halt.
;;;
define zrprep2;
vars n;
if zaentry(6)=[no] and not(zejoint or zejoint=1) then return(zagoto(9)); endif;
.zaindex->n;
nil,nil,nil,nil,nil,nil->zpgoaltype->zpgoal->zptree->zpstate->zpactor->zpmgoal;
sub("zpmgoal",n,zkgoal);sub("zpgoal",n,zkgoal);sub("zptree",0,[%n%]);sub("zpstate",n,notyet);
sub("zpactor",n,both);sub("zpgoaltype",n,"plan");
true->zejoint;
zaenter1("zrbasic",1,zkgoal);zagoto(8);
enddefine;
;;;
;;;         9. Keeps swapping. This is just to tell chairman that the goal
;;;         has been achieved and one of the robots wants to stop talking.
;;;
define zrhalt;
if zaentry(4)=[achieved] or zaentry(6)=[no] then npr(jtopics);npr(mtopics);stop+1->count;endif;
enddefine;
```

```
*** ROUTINE. ACHIEVE.p (1 of 3) ***
;;;
;;;         1. Enters next goal to be tackled.(Power 1974,p279)
;;;
;;;
define zrcurrgl;
vars g;
zpnextgl()->zpcurr;
zpgoal,zpcurr.suc->g;
zaenter(g);
zaput(goaltype,zpgoaltype,zpcurr.suc);
enddefine;
;;;
;;;         2. Enters actor for current goal.
;;;
define zractor;
(zpactor,zpcurr.suc :: nil).zaenter;
enddefine;

;;;
;;;         3. Enters state of goal. asks if necessary.
;;;
define zrstate;
vars a b n s;
[]->a;
zaentry(1)->b;zpparent(zpcurr)->n;  zpgoal,n.suc-> s;
if zatake(goaltype)="plan" then
  if zcevent(b) then
   if zehold=[] then
     zkstate(s)  else
     zaenter([notyet]);
    endif;
  else
   zkstate(b);
  endif;
else
   zaenter([notyet]);
endif;
enddefine;
```

```
*** ROUTINE.ACHIEVE.p (2 of 3) ***
;;;
;;;         4. Enters kind of goal: [event] or [sitn];
;;;            and branch to planning or execution.
;;;
define zrkind;
vars a b n s;
zaentry(1)->b;zpparent(zpcurr)->n;  zpgoal,n.suc-> s;
zaentry(3)->a;
if ispair(a) then sub("zpstate",zpcurr,hd(a)) endif;
if a=[achieved] then
   zpdelete(zpcurr);zpchop(zpcurr);.zpprune;
   .zeexit;
    if zptree=[] then
     .zejump;zagoto(8); zaenter(a)
    else
     zeload("zrachieve");zaput(goaltype,"plan");
    endif;
endif;
if zcevent(zaentry(1)) then
  if a=[achieved] and suc3(zkworld,hd(s))="seen" then
    if suc(zkxsee,hd(s))=true or suc(zkxsee,hd(s))=1 then else
       if zehold=[] and zkinfoasked(s)/=[asked]
         then zeplay("zgtell",[1 is]<>s);endif;
    endif;
  endif;
else
  if a=[achieved] and suc3(zkworld,hd(b))="seen" then
    if suc(zkxsee,hd(b))=true or suc(zkxsee,hd(b))=1 then else
      if zehold=[] and zkinfoasked(b)/=[asked]
        then  zeplay("zgtell",[1 is]<>b);endif;
    endif;
  endif;
endif;
if zcevent(zaentry(1)) then
      zaenter([event]);
      if a=[notyet] then
         .zeexit;zeload("zrassess");
      endif;
else   if a=[notyet] then
      zaenter([sitn]);endif;
endif;
enddefine;
;;;
;;;         5.  Sets up routine or game to find plan. entry is a plan or [no].
;;;             (Power p280)
;;;
define zrplan1;
if zaentry(3)/=[notyet] then return(zagoto(7)) endif;
if zaentry(2)=[both] then
  if zatake(goaltype)="plan" then zeplay("zgplan",undef)
  elseif zatake(goaltype)="experiment" then
   zeplay("zgexperiment",undef)
  else zeplay("zgtest",zaentry(1));
  endif;
else
zeload("zrplan");
endif;
enddefine;
```

```
*** ROUTINE. ACHIEVE.p (3 of 3) ***
;;;
;;;         6. Examines plan in tree; fails the goal if no plan was found,
;;;         or reloads routine for iterative action Q
;;;         expand plan to do preconditions, or execution of plan.
;;;
define zrreturn;
vars p gt;
zatake(goaltype)->gt;
zaentry(5)->p;
if p=[no] and gt/="plan" then
  return(sub("zpstate",hd(zptree,0.suc),failed),zagoto(7))
endif;
if p=[no] then
  zphang([both [undef]],zpnextgl(),"experiment")
else
  zphang(p,zpnextgl(),gt);
endif;
zagoto(7);
enddefine;
;;;
;;;         7. Prune tree. Function defined  as for Assess.
;;;
;;;
;;;          8. Prune control stack. Function defined as for Assess.
;;;
```

```
*** ROUTINE.PLAN.p (1 of 2) ***
 ;;;
 ;;;
 ;;;        1 Enters goal for which plan is needed.
 ;;;
 define zrgoal();
zpgoal,zpcurr.suc.zaenter;
zaput(goaltype,zpgoaltype,zpcurr.suc);
enddefine;
 ;;;
 ;;;        2 Finds rule to achieve required result: if none fails.
 ;;;        If desperate, use rules with undef as result.
 ;;;        If still desperate, use rules with nothing as result.
 ;;;
define zrrule();
 vars r o;
zctypof(hd(zaentry(1)))->o;
if not(zkfluid() or zkfluid()=1) then zeexit();return(zaenter([no]));endif;
if zatake(goaltype)="plan" or zatake(goaltype)="experiment" then
  zkres([%o%])->r;
endif;
if zatake(goaltype)="test" then zksit(o::tl(zaentry(1)))->r;endif;
zaenter(r);
if r/=undef then  zkdel(zkrules,suc(r,evt))<>[%r%]->zkrules;endif;
enddefine;
 ;;;
 ;;;        3 Decides who does the action, and enters a specific rule.
 ;;;
define zrspec();
 vars r,e,a;
 zaentry(2)->r;
 if r=undef then return(.zeexit,zaenter([no]),zaput(expectreply,true));endif;
 zkme::(tl(suc(r,evt)))->e;
 ;;;
 ;;;        No point trying to do it yourself if your partner can do it more
 ;;;        easily - CGB.
 ;;;
 if suc(r,res)=[robot] then (hd(zaentry(1))::tl(e))->e;endif;
 if hd(suc(r,evt))="robot" and hd(suc(r,sit))="robot" then
   if zktval(zkme::(tl(suc(r,sit))),zkworld)=true then  zkme::(tl(suc(r,evt)))->e; endif;
   if zktval(zkyou::(tl(suc(r,sit))),zkworld)=true then zkyou::(tl(suc(r,evt)))->e;endif;
 endif;
 if hd(e)=zkme then goto me else goto him endif;
;;;
;;;         this was a bug 'cos it assumes zkme can do the event under the
;;;         constraints of the situation. See note on zkcan.
;;;
 me:
 if zkcan(suc(zksrule(e),evt)) then
  if suc(zksrule(e),sit)/=[any] and zatake(goaltype)/="test" then
   if zkis(suc(zksrule(e),sit))/=[asked]
   then return(zaenter(zkspec(r,e)));else
   return(zawipe("zrplan"));
   endif;
 else return(zaenter(zkspec(r,e)));
 endif;
endif;
 him: if zejoint
 then zkyou::tl(e)->e;zkcan(e)->a;
 if a=[asked] then return;
 elseif a=1 or a=true then return(zaenter(zkspec(r,e)));
else return(.zeexit,zaenter([no]),zaput(expectreply,true))
 endif;
else .zeexit;zaenter([no]);
endif;
enddefine;
```

```
*** ROUTINE.PLAN.p (2 of 2)**
 ;;;
 ;;;
 ;;;        4 Enters the state of the sit, asking if necessary.
 ;;;
define  zrsit;
 vars s a;
if zatake(goaltype)="plan" then
  suc(zaentry(3),sit)->s;
  if s=[any] then zaenter([achieved]) else
   if zkinfoasked(s)/=[asked] then zkstate(s) else
    if zkis1(s)=undef then zaenter([failed]) endif;
    if zkis1(s)=false or zkis1(s)=0 then zaenter([notyet]) endif;
    if zkis1(s)=true  or zkis1(s)=1 then zaenter([achieved]) endif;
   endif;
  endif;
elseif zatake(goaltype)="test" then
   suc(zaentry(3),res)->s;zaenter([achieved]);
   if zkinfoasked(tl(zamakeq(hd(s))))/=[asked] then zkis(tl(zamakeq(hd(s))));
   else if zkis1(tl(zamakeq(hd(s))))=undef then zaenter([failed]);endif;
   endif;
elseif zatake(goaltype)="experiment" then
   suc(zaentry(3),evt)->s;suc(zctobjs,hd(tl(s)))->s;
   if s=[robot] then [^(hd(suc(zaentry(3),evt)))]->s;endif;
   zaenter([achieved]);
   if zkinfoasked(tl(zamakeq(hd(s))))/=[asked] then zkis(tl(zamakeq(hd(s))));
   else if zkis1(tl(zamakeq(hd(s))))=undef then zaenter([failed])endif;
   endif;
endif;
enddefine;
 ;;;
 ;;;        5. Completes plan and enters in the calling f/g.
 ;;;
define zrcomp;
vars p,r;
if zaentry(4)=[failed] then [no]->p;
else
          zaentry(3)->r; nil->p;
          if zaentry(4)=[notyet]then
                    [%suc(r,sit)%]->p;
                    if zejoint or zejoint=1 then both else zkme endif; ::p->p;
          endif;

          suc(r,evt) -> r;  p<>[%r.hd, r%] -> p;
endif;
.zeexit;zaenter(p);zaput(expectreply,true);
enddefine;
```

```
*** ROUTINE.FINDOUT.p (1 of 1) ***
;;;
;;;
;;;          load information that is required or the goal.
;;;
define zrinformation;
zaenter(zemove1);
enddefine;
;;;
;;;          Unless your partner doesn't know ask him.
;;;
define zrask;
vars a;
zaentry(1)->a;
;;;  if suc(zkxsee,hd(a))/=0 and suc(zkxsee,hd(a))/=false then
   zeplay("zgask",[is]<>a);
;;; else
;;;    zaenter([undef]);
;;; endif;
enddefine;
;;;
;;;          If you both don't know then do a test.
;;;
define zrtry;
vars a;
zaentry(2)->a;
if hd(a)=undef then
if zkis1(a)=true or zkis1(a)=1 then zaenter(a);return;
elseif zkis1(a)=false or zkis1(a)=0 then zaenter(zkopp(a));return;
endif;       zaput(goaltype,"test");
         zphang([both ^(zaentry(1))],zpnextgl(),"test");
         zeload("zrachieve");
         return;
else
  zaenter(a)
endif;
enddefine;
;;;
;;;          Conclude.
;;;
define zrconclude;
vars a;
zpnextgl()->zpcurr;
if zaentry(2)=[undef] and zaentry(3)=[undef] then  [failed]->a;endif;
if zaentry(2)=[undef] and zaentry(3)/=[undef] then
  if zaentry(3)/=zaentry(1) then
    [notyet]->a;
    else
    [achieved]->a;
  endif;
endif;
if zaentry(2)/=[undef] then
  if zaentry(2)/=zaentry(1) then
    [notyet]->a;
     else
    [achieved]->a;
  endif;
endif;
.zeexit;
if zatake(kind)="routine" and zatake(place)/=9 and zatake(goaltype)/="experiment" and zatake(goaltype)/="test" then
zaenter(a);
endif;
if zatake(place)=12 then zaenter(a);endif;
enddefine;
```

```
*** ROUTINE.ASSESS.p (1 of 5) ***
;;;
;;;
;;;         1. Enter intended goal that the action was supposed to achieve if there
;;;         is one.
;;;
define zrgoalintended;
if zatake(goaltype)="none" then return(zaenter([undef]));endif;
zpnextgl()->zpcurr;
zaenter(zpgoal,zpparent(zpcurr).suc);
enddefine;
;;;
;;;         2. Enter action, in the case of an unepected event enter undef.
;;;
define zraction;
if zatake(goaltype)="none"
then return(zaenter([undef]));endif;
zaenter(zpgoal,zpcurr.suc);
zaput(goaltype,zpgoaltype,zpcurr.suc);
enddefine;
;;;
;;;         3. Enter actor as defined in Achieve.
;;;
;;;
;;;         4. Enters precondition for action to succeed. Entering undef when the
;;;         the goal is undefined.
;;;
define zrprecondition;
if zatake(goaltype)="plan" then
  zaenter(suc(zkres([^(zctypof(hd(zaentry(1))))]),sit));
  return;
endif;
if zatake(goaltype)="test" then
  zaenter(zaentry(1));
  return;
endif;
if zatake(goaltype)="experiment" then
  zaenter([any]);
  return;
endif;
if zatake(goaltype)="none" then
  zaenter([undef]);
  return;
endif;
enddefine;
;;;
;;;         5. Enters object of action that was supposed to be under examination.
;;;
define zrobject;
if zatake(goaltype)="plan" then
  zaenter(hd(zaentry(1))::nil);
endif;
if zatake(goaltype)="test" then
  zaenter(suc(zksit(zaentry(1)),res));
endif;
if zatake(goaltype)="experiment" then
  zaenter(suc(zctobjs,hd(tl(zaentry(2)))));
endif;
if zatake(goaltype)="none" then
  zaenter([undef]);
endif;
if zaentry(5)=[robot] then zaenter(zaentry(3));endif;
if suc3(zkworld,hd(zaentry(5)))="estimated"
then sub2("zkworld",hd(zaentry(5)),wpos(hd(zaentry(5))),undef);endif;
enddefine;
```

```
*** ROUTINE.ASSESS.p (2 of 5) ***
;;;
;;;           6.  Enters rule used to achieve intended goal.
;;;
define zrrules;
vars p;
zctypof(hd(zaentry(2)))::tl(zaentry(2))->p;
if zatake(goaltype)="plan" then
  zaenter(zkevt(p));
  return;
endif;
if zatake(goaltype)="test" then
  zaenter(zkevt(p));
  return;
endif;
if zatake(goaltype)="experiment" then
  zaenter(zkevt(p));
  return;
endif;
if zatake(goaltype)="none" then
  zaenter([undef]);
  return;
endif;
enddefine;
;;;
;;;          7. Enters current value of world model for later comparison.
;;;
define zrbefore;
zaenter(zkworld);
[GOAL ^(zaentry(1)) GOALTYPE ^(zatake(goaltype)) ACTOR ^zkme  NAME ^(zatake(name)) EFFECTS unknown  ITEMS
^zginformation STATUS open]::ztopics->ztopics;
enddefine;
;;;
;;;          8.  Whoever is to do the action tells the other when it is done
;;;
define zract;
;;;if zatake(goaltype)/="none" then
 []->zginformation;  ;;; Initialise agenda of things "said" and "to say".
;;;endif;
if zatake(goaltype)="none" then return(zaenter([done]),.zklook);endif;
if zaentry(3)=[%zkyou%] then
return(zepost([--])) endif;
zpdo(zaentry(2));
.zklook;
zaenter([done]);
sub("zpstate",zpcurr,achieved);
if zejoint or zejoint=1 and zehold=[] then zeplay("zgtell",zaentry(2)) endif;
enddefine;
```

```
;;;
;;;         9.  Enters name of object which changed position, or [nothing] if
;;;         none did.
;;;
define zrresult;
vars d;
if hd(zaentry(5))/=undef and suc3(zkworld,hd(zaentry(5)))/="seen"  and
zkinfoasked(tl(zamakeq(hd(zaentry(5)))))/=[asked] then
zkis(tl(zamakeq(hd(zaentry(5)))))->d;
if d=[asked] then return;endif;
endif;
zadiff(zaentry(7),zkworld)->d;
if hd(d)=undef then [nothing]->d;endif;
if d/=[nothing] and (zejoint or zejoint=1) and zehold=[]
              and zkinfoasked(d)/=[asked] then
   zeplay("zgtell",d);return;
 endif;
if d=[nothing] and hd(zaentry(5))/=undef and
           suc3(zkworld,hd(zaentry(5)))="seen" and
           (zejoint or zejoint=1) and zehold=[] and
           zkinfoasked(d)/=[asked] then
   zeplay("zgtell",d);return;
endif;
if d=zaentry(5) or zaentry(5)=[undef] then
   zaenter(d);
   else
   zaenter([nothing]);
endif;
if d/=[nothing] then
     [^(hd(d)) ^(suc2(zkworld,d.hd))]->d;
     if (zejoint or zejoint=1) and zehold=[] and zkinfoasked(d)/=[asked] then
       zeplay("zgtell",[1 is]<>d);return;
     endif;
endif;
enddefine;
;;;
;;;         10. finds state of parent goal and puts state on tree.
;;;         (Power 1974 p281)
;;;
define zrparent;
vars a,s,p,e,n;
if zatake(goaltype)="none" then return(zaenter([achieved]));endif;
zpparent(zpcurr)->n; ;;; Goal number
zaentry(1)->s;       ;;; Current goal
zaentry(4)->p;       ;;; Precondition
zaentry(5)->e;       ;;; effect
if hd(s)/=undef then
 if s=p then
   zaenter([achieved]);
 else
   zkstate(s);              ;;; calc. and enter state of parent's goal.
 endif;
else
 zaenter([achieved]);
endif;
zaentry(10)->a;                ;;; fetch copy of result
if ispair(a) then sub("zpstate",n,hd(a)) endif; ;;; enter it in globals.
if s/=[undef]  and s/=p and a=[achieved] and zkinfoasked(s)/=[asked] then
    zeplay("zgtell",[1 is]<>s)
endif;
enddefine;
define zrparent;
zaenter([achieved]);
enddefine;
```

```
*** ROUTINE.ASSESS.p (4 of 5)
;;;
;;;         11. Learns lesson from seeing result of event. If partner is known to
;;;         believe a rule which predicts the wrong result, teaches him a better rule.
;;;
define zrlesson;
vars p s a o;
zaentry(2)->a;    ;;; Action
zaentry(1)->s;    ;;; Current Goal
zaentry(4)->p;    ;;; Precondition for goal to succeed
zaentry(5)->o;    ;;; Object of action under investigation
;;;
;;;         First inference in which the action is not known.
;;;
if a=[undef] then
  if zaentry(9)/=[nothing] then
  hd(zaentry(9))::[^(suc2(zkworld,hd(zaentry(9))))]->s;
  if zehold=[] and zkinfoasked(s)/=[asked] then zeplay("zgtell",[1 is]<>s);return;endif;
if zaentry(9).hd.zctypof="robot" then
  [^(hd(zaentry(9))) ^(pre(zctobjs,[^(zctypof(hd(zaentry(9))))]))]->a;
else
  [robot ^(pre(zctobjs,[^(zctypof(hd(zaentry(9))))]))]->a;
endif;
  zaenter([learned]);
  if zehold=[] and zkinfoasked(a)/=[asked] then zeplay("zgtell",a);endif;
  return;
  else
  return(zaenter([nothinglearned]));
  endif;
else
;;;
;;;         Second inference in which the precondition is not known, there is an action and we know it's effect
;;;
   if a/=[undef] and  p/=[any] and suc3(zkworld,p.hd)/="seen" and
     (suc3(zkworld,o.hd)="seen" or suc3(zkworld,o.hd)="told" or
      suc3(zkworld,o.hd)="inferred") then
   if zaentry(10)=[achieved] then
     zaenter([learned]);
     if zaentry(9)=[nothing] then
     zkopp(s)->s;
     endif;
     sub2("zkworld",hd(s),hd(tl(s)),"inferred");
     if zehold=[] and zkinfoasked(s)/=[asked] then zeplay("zgtell",[1 is]<>s);endif;
    else
    return(zaenter([nothinglearned]));
     endif;
  return;
  endif;
endif;
;;;
;;;         Third and fourth inferences in which there is an action, the precondition
;;;         is known but we don't know it's effect
;;;
if  zaentry(1)=[undef] or (zaentry(2)/=[undef] and (suc3(zkworld,p.hd)="seen"
or suc3(zkworld,p.hd)="told" or suc3(zkworld,p.hd)="inferred"))
then
if zkpred(a,zaentry(7))/=zaentry(9)
then zkponder(a,zaentry(9))->p; if p=[asked] then return endif;
endif;
zaenter([learned]);zkxpred(a,zaentry(7))->p;
if p/=undef and p/=zaentry(9) and zkinfoasked(zkrule(a))/=[asked]
then zeplay("zgrule",zkrule(a))
endif;
endif;
enddefine;
```

```
*** ROUTINE.ASSESS.p (5 of 5) ***
;;;
;;;         12. Prunes plan tree up to the highest achieved goal.Based on what has
;;;          been said and what goals are known to be true and also what one
;;;          believes to be true privately.
;;;
define zrprune;
vars l p s;10000->p;
zpsearch(zpstate,"achieved",[any])->s;
if zatake(name)/="zrachieve" then
   zpworkthrupublic(zpgoal,zginformation)->p;
endif;
zpworkthruprivate(zpgoal,zkworld)->l;
if s<p then s else p endif;->p;
if l<p then l else p endif;->p;
if zatake(name)="zrachieve" and zatake1("zrassess",place)/=undef and
                        zatake1("zrassess",place)>=8 then
   10000->p;
endif;
;;; don't prune if you're in the middle of assessing something.
if p<=zpnextgl() then
   zaenter(zplist(p));zpdelete(p);zpchop(p);.zpprune;
   zpnextgl()->zpcurr;npr(zaentry(12));
else
   zaenter([none]);
endif;
enddefine;
;;;
;;;         13. Prunes control stack for conversational procedures that relate
;;;         to plans that are now achieved.
;;;
define zrclean;
vars p;
zaentry(zatake(place)-1)->p;
[GOAL ^(zaentry(1)) GOALTYPE ^(zatake(goaltype)) ACTOR ^zkme  NAME ^(zatake(name)) EFFECTS ^p  ITEMS
^zginformation STATUS closed]::ztopics->ztopics;
if zptree=[] then
   .zejump;zagoto(8);zaenter([achieved]);return;
endif;
if p/=[none] then
   zaworkthru(p)->p;
endif;
if p/=[none] then
   zakillafter(p);
else if zatake(name)="zrachieve" then
   .zeexit;
   zeload("zrachieve");
   else
   .zeexit;
   endif;
endif;
enddefine;
```

```
*** GAMES.p (1 of 1)***
;;;
;;;         (Power 1974 p284) GAMES Macros
;;;
game zggame;
1. * w zgname [zdgame];
2. * b zgready [zdsign];
3. ^ w zgload;
end;

game zgask;
1. * w zgquery [zdquery];
2. * b zganswer [zdsign];
3. ^ w zgrecord;
end;

game zgtell;
1. * w zgrelate [zdfact zdevent zdobj];
2. * b zgexamine [zdsign];
3. ^ w zgrelook;
end;

game zgrule;
1. * w zgwrule [zdrule];
2. * b zgbreply [zdsign];
3. * w zgwnote ;
4. * b zgbrule [zdrule];
5. * w zgwreply [zdsign];
6. ^ b zgbnote;
end;

game zggoal;
1. * w zgplead [zdsitn];
2. * b zgreact [zdsign];
3. ^ w zgreport;
end;

game zgplan;
1. * w zgsuggest [zdplan zdsign];
2. * b zgrespond [zdsign];
3. ^ w zgreturn;
end;

game zgexperiment;
1. * w zgsuggest [zdplan zdsign];
2. * b zgrespond [zdsign];
3. ^ w zgreturn;
end;

game zgtest;
1. * w zgsuggest [zdplan zdsign];
2. * b zgrespond [zdsign];
3. ^ w zgreturn;
end;
```

*** GAME.GAME.p (1 of 1) ***

```
;;;         Used to get another game loaded. can be used anywhere.
;;;         (dot notation used in all game routines) (Power 1974 p285).

;;;
;;;         White names the game.
;;;
define zgname;
zaput(expectreply,true);
zegame1.zaenter;
enddefine;
;;;
;;;         Black agrees to play, possibly after a delay while he reaches a
;;;         suitable point in his routines, loads the game taking black and
;;;         exits from mggame.
;;;
define zgready;
vars g;1.zaentry->g;
if g=[jump] then g.zaenter;return(.zejump) endif;
if g=[zggoal] and zatake1("zrbasic",place)=1 then
         zemark([zrbasic 3]);return
         elseif (g=[zgplan] or g=[zgtest] or g=[zgexperiment])
        and zatake1("zrachieve",place)/=5
then return([zrachieve 5].zemark) endif;
if memb(hd(g),[zgrule zgtell])
and suc(zecontrol.tl.hd,kind)="game"
then zawipe(suc(zecontrol.tl.hd,name))
endif;
false->z_i_have_tried;[yes].zaenter;.zeexit; g.hd.zeload; zaput(colour,black);
zaput(expectreply,true);
enddefine;
;;;
;;;         White loads the game, taking white, and exits.
;;;
define zgload;
vars g; 1.zaentry->g;
if g=[jump] then .zejump   else .zeexit;false->z_i_have_tried; g.hd.zeload endif;
enddefine;
```

```
*** GAME.ASK.p (1 of 1) ***
;;;
;;;         Used anywhere. asks either can or is questions.
;;;         learns not only the answer, but whether or not the other
;;;         robot knows the answer.
;;;          (Power 1974, p286)
;;;
;;;         1. Asks question.
;;;
define zgquery;
zaput(expectreply,true);
zemove1.zaenter;
enddefine;
;;;
;;;         2. Tries to find answer, using own resources only. if it is an
;;;         "is question, records the fact that white didnt know the answer.
;;;         Need to record that although white didn't know it he does now.CGB.
;;;
define zganswer;
vars q g;
1.zaentry->q;tl(q)->g;
if q.hd="is"
then sub("zkxsee",q.tl.hd,0);q.tl.zkis1
else q.tl.zkcan1
endif;->q;
[%q.zoyn%].zaenter;
.zeexit;if q=undef then
 sub2("zkworld",g.hd,suc(zkworld,g.hd),"asked");
 zeload("zrfindout");zaenter(g);
zagoto(2);zaenter([undef]);.zrtry;endif;
enddefine;
;;;
;;;         3. Records the answer and whether black knew it.
;;;
define zgrecord;
vars q,a,g;
1.zaentry->q;
2.zaentry->a;
if a=[undef] then undef else (a=[yes]) endif;->a;
if q.hd="can" then sub("zkxacts",q.tl.tl.hd,a);return(.zeexit) endif;
;;;         this does the following:
;;;         if your partner doesnt know then it's time to do a test
;;;         find a rule in which the precondition is true
;;;         and is the same as what you are asking
;;;          find it's corresponding effect call it result.
;;;          find the action
;;;          decide who's to do it
;;;         perform it
q.tl->q;sub("zkxsee",q.hd,(a/=undef));
if a=0 or a=false then q.zkopp->q endif;
if a/=undef then sub2("zkworld",q.hd,q.tl.hd,"told"); endif;
.zeexit;if a=undef then sub2("zkworld",q.hd,suc(zkworld,q.hd),"asked");zaenter([undef]);.zrtry; else zaenter(q);endif;
enddefine;
```

```
*** GAME.TELL.p (1 of 2) ***
;;;     (Power 1974, p287)
;;;
;;;             White enters a fact.
;;;
define zgrelate;
zaput(expectreply,true);
zemove1.zaenter;
enddefine;
;;;
;;;             Black updates his models of object positions, and of white,
;;;             and responds to what white told him.
;;;
define zgexamine;
vars d,f;1.zaentry->f;
if zatake1("zrassess",place)=undef then
    [zrassess 1].zemark;zeload("zrassess");
    zaput(goaltype,"none");zerevert();
endif;
if zcobject(f)
then  if zatake1("zrassess",place)>=10
    then
    [undef].zaenter;.zeexit;
    if f/=[nothing] then zagoto(9);f.zaenter;zagoto(10);endif;
    else
    [zrassess 10].zemark;
    while zatake(name)/="zrassess" do tl(zecontrol)->zecontrol endwhile;
    endif;
return();
endif;
if zcevent(f)
then if zatake1("zrassess",place)=8
   then
   []->zginformation;
       if zatake1("zrassess",goaltype)/="none" then sub("zkxacts",f.tl.hd,1); sub("zpstate",zpcurr,achieved);endif;
   [undef].zaenter;.zeexit;[done].zaenter; .zklook;
   else if zatake1("zrassess",place)>8 then  [zrassess 1].zemark;zeload("zrassess");
zaput(goaltype,"none");zerevert();endif;
   [zrassess 8].zemark;
   while zatake(name)/="zrassess" do tl(zecontrol)->zecontrol endwhile;
   endif;
 return();
endif;
if f.tl.hd="is"
then if f.hd=1 or f.hd=true then f.tl.tl else f.tl.tl.zkopp endif;->f;
   if memb(f.hd,zksee)
   then if f.zkis1
       then [yes];if suc(zkxsee,f.hd)/=0 and suc(zkxsee,f.hd)/=false then
       sub("zkxsee",f.hd,1);endif;
       else [no];sub("zkxsee",f.hd,0);
       endif;
   else [undef];if suc(zkxsee,f.hd)/=0 and suc(zkxsee,f.hd)/=false then
       sub("zkxsee",f.hd,1);endif;
sub2("zkworld",f.hd,f.tl.hd,"told")
   endif;
else if f.tl.tl.hd=zkme
   then if memb(f.tl.tl.tl.hd,zkacts)=f.hd
       then [yes]
       else [no]
       endif;
   else [undef]; sub("zkxacts",f.tl.tl.tl.hd,f.hd);
endif;
endif;
.zaenter;.zeexit;
if f=suc(zpgoal,zpcurr) then zawipe("zrachieve");endif;
enddefine;
```

```
*** GAME.TELL.p (2 of 2) ***
;;;
;;;         White updates his models of object positions, and of black, in
;;;         the light of blacks response.
;;;
define zgrelook;
vars f,a;1.zaentry->f;2.zaentry->a;
if f.hd.zcobject then return(.zeexit) endif;
if f.zcevent then return(.zeexit) endif;
if f.tl.hd="is"
then if a=[no]
   then sub("zkxsee",f.tl.tl.hd,not(memb(f.tl.tl.hd,zksee)));
       if suc(zkxsee,f.tl.tl.hd)
       then if f.hd=0 then f.tl.tl else f.tl.tl.zkopp endif;
           ->f;sub2("zkworld",f.hd,f.tl.hd,"told")
       endif;
    else sub("zkxsee",f.tl.tl.hd,(a=[yes]))
    endif;
else if f.tl.tl.hd=zkyou and a=[no]
   then sub("zkxacts",f.tl.tl.tl.hd,f.hd.not)
   endif;
endif;
.zeexit;
enddefine;
```

```
*** GAME.RULE.p (1 of 1) ***
;;;
;;;         Used to explain, or compare views on, the rules which specify the
;;;         consequences of actions. can be played anywhere.
;;;          (Power 1974, p289)

;;;
;;;         White announces a rule that he wants black to adopt
;;;
define zgwrule;
zaput(expectreply,true);
zemove1.zaenter;
enddefine;
;;;
;;;          Black agrees, exiting the game if he agrees.
;;;
define zgbreply;
vars r1,r2,b;
1.zaentry->r1; r1,evt.suc.zkrule->r2;
r1.zkxadd; zkbetter(r1,r2)->b;
if zaequal(r1,r2) then [yes].zaenter;.zeexit
elseif b=undef then r1.zkxadd;[undef].zaenter;.zeexit
elseif b=true or b=1 then r1.zkadd; [undef].zaenter;.zeexit
else zaput(expectreply,true);[no].zaenter;
endif;
enddefine;

;;;
;;;          If black agreed, white updates his model of blacks rules and exits.
;;;          If not, he goes to 4 and allows black to announce his rule.
;;;
define zgwnote;
if 2.zaentry=[no] then zepost([--]);zaput(expectreply,true);
4.zagoto else 1.zaentry.zkxadd;.zeexit endif;
enddefine;
;;;
;;;           Black announces his version of the rule white announced at 1.
;;;
define zgbrule;
zaput(expectreply,true);
1.zaentry,evt.suc.zkrule.zaenter;
enddefine;
;;;
;;;         White responds, updates his world model in a way depending on
;;;         whose rule was better, and exits.
;;;
define zgwreply;
vars r1,r2,b;
4.zaentry->r1;
1.zaentry->r2;
r1.zkxadd;zkbetter(r1,r2)->b; if b=true or b=1 then r1.zkadd endif;
if b=undef and not(zaequal(r1,r2)) then false->b endif;
if b=undef then [yes] elseif b=true or b=1 then [undef] else [no] endif;
.zaenter;.zeexit;
enddefine;
;;;
;;;          Black updates his model of whites rules, then exits.
;;;
define zgbnote;
if 5.zaentry=[undef] then 4.zaentry.zkxadd endif;.zeexit;
enddefine;
```

```
*** GAME.GOAL.p (1 of 1) ***
;;;
;;;         Played with white at zrbasic 6 and black anywhere
;;;

;;;
;;;          1. White announces his goal
;;;
define zgplead;
zaput(expectreply,true);
zemove1.zaenter;
enddefine;
;;;
;;;         2. If black has a different goal he refuses; if not, he accepts
;;;         and goes to zrbasic 7.
;;;
define zgreact;
1.zaentry->zkxgoal;
if zkgoal=[none] then zkxgoal->zkgoal endif;
if zkgoal=zkxgoal then [yes] else [no] endif; .zaenter;
if zkgoal=zkxgoal then true->zejoint;return(.zejump) endif;
.zeexit;
enddefine;
;;;
;;;         3. Updates model of partners goal, and enters his reply
;;;         in zrbasic 6.
;;;
define zgreport;
vars r; 2.zaentry->r;
if r=[yes] then zkgoal->zkxgoal;true->zejoint endif;
.zeexit;r.zaenter;
enddefine;
```

```
*** GAME.PLAN.p (1 of 1) ***
;;;
;;;          Played with both robots at zrachieve 5. used to agree on a plan
;;;         experiment or Instrumental test.
;;;          (Power 1974, p292)
;;;
;;;          White loads zrplan, which makes and enters the plan.
;;;
define zgsuggest;
"zrplan".zeload;
enddefine;
;;;
;;;         Black judges the plan and either agrees with it and enters it in
;;;         zrachieve 5, or arranges zgrule to explain the rule by which he
;;;          rejects it.
;;;
define zgrespond;
vars p,a,n;
zatake(name)->n;
1.zaentry->p;
if suc(hd(tl(zecontrol)),name)/="zrachieve" then
 if suc(hd(tl(zecontrol)),place)/=5  then
zemark([zrachieve 5]);return;
endif;endif;
if p=[no]
then
    if .zkfluid and z_i_have_tried=false
    then [yes].zaenter;.zeexit;n.zeload
    else [no].zaenter;.zeexit;[no].zaenter;6.zagoto
    endif;false->z_i_have_tried;
else if p.zpevt.zkcan1=false
    then
     [%(zpcurr+1)::p.zpevt%]<>zpmgoal->zpmgoal;
    zeplay("zgtell",[0 can]<>p.zpevt);n.zawipe;
    endif;
    zkjudge(p,suc(zpgoal,zpnextgl()))->a;
    if a=[asked] then return() endif;
    if (a=0 or a=false) and zkbetter(p.zpevt.zkrule,p.zpevt.zkxrule)=true
    and (suc(p.zpevt.zkrule,res)/=[undef]
        or (zatake(name)="zgtest" and
        suc(p.zpevt.zkrule,res)=[undef])
)
    then p.zpevt.zkrule->a;
        if suc(a,res)=[nothing] then a.zkinvert->a endif;
        [%(zpcurr+1)::a%]<>zpmgoal->zpmgoal;
        zeplay("zgrule",a);n.zawipe
    else [yes].zaenter;.zeexit;p.zaenter;6.zagoto
    endif;
endif;
enddefine;
;;;
;;;         White makes an entry in zrachieve 5, and exits.
;;;
define zgreturn;
vars p n;
zatake(name)->n;
1.zaentry->p;
if p=[no] and 2.zaentry=[yes]
then true->z_i_have_tried;.zeexit;n.zeload;
zaput(colour,black);zaput(expectreply,true);
else false->z_i_have_tried;.zeexit;5.zagoto;p.zaenter;6.zagoto;
endif;
enddefine;
```

```
*** AUXFUNS.p (1 of 4) ***
;;;
;;;
;;;          (Power 1974, p295)
;;;
;;;


;;;
;;;          Puts x after n in routine/game w in control
;;;
;;;
define zaput1(w,n,x);
vars l1,l2,l3;
zecontrol->l2;
nil->l1;
loop:
if null(l2) then return endif;
hd(l2)->l3;
if suc(l3,name)=w
then rep(l3,n,x)->l3;
rev(l1)<>(l3::tl(l2))->zecontrol
else l3::l1->l1;tl(l2)->l2; goto loop
endif;
enddefine;

define zatake1 (w,n);
vars l1;
zecontrol->l1;
while ispair(l1) and suc(hd(l1),name)/=w then tl(l1)->l1 endwhile;
if null(l1) then undef else suc(hd(l1),n) endif;
enddefine;

;;;
;;;          Finds item after n in routine/game w in control.
;;;
;;;
;;;
;;;          Puts in x at n in entries of w.
;;;
define zaenter1(w,n,x);
zaput1(w,entries,rep(zatake1(w,entries),n,x));
enddefine;
;;;
;;;          Finds entry at n in routine/game w.
;;;

define zaentry1(w,n);
suc(zatake1(w,entries),n);
enddefine;

;;;
;;;          Finds item after n in top r/g of control.
;;;
define zatake(n);
suc(hd(zecontrol),n);
enddefine;

;;;
;;;          Puts x after n in top r/g of control.
;;;
define zaput(n,x);
zaput1(zatake(name),n,x);
enddefine;
```

```
*** AUXFUNS.p (2 of 4) ***
;;;
;;;
;;;        (Power 1974, p298) Arranges for entry x to be translated into english
;;;        then posts it in zebox.

;;;        Changes the name of z?name... by substituting 'w' for g.
;;;
define zautter(x);
vars n;
if zatake(kind)="routine" then return;endif;
valof(zatake(name))->n;
if (n,place.zatake.suc.tl.hd/=zatake(colour)) then return;endif;
[% destword(hd(tl(tl(suc(n, zatake(place)))))) %]->n;
(hd(n) :: ( (`w`)  :: tl(tl(n)))) ->n;
while ispair(n) do hd(n);tl(n)->n endwhile;consword()->n;
;;;zepost(popval([%x,".",n%]));
[%x,".",n%] -> n;
popval(n) -> n;  zepost(n);
enddefine;


;;;
;;;        Enters x at current place in top r/g
;;;
define zaenter(x);
if pro then zosp();pr(zatake(place));sp(1);pr(x);
prs(' entered by ');pr(zkme); endif;
zaenter1(zatake(name),zatake(place),x);zautter(x);
enddefine;
;;;
;;;        Finds entry at place n in top r/g
;;;
define zaentry(n);
zaentry1(zatake(name),n);
enddefine;
;;;
;;;        Kills any r/g named w.
;;;
define zakill(w);
vars l1;
nil->l1;
while ispair(zecontrol) and zatake(name)/=w do
hd(zecontrol)::l1->l1;tl(zecontrol)->zecontrol endwhile;
if ispair(zecontrol) then tl(zecontrol)->zecontrol endif;
rev(l1)<>zecontrol->zecontrol;
enddefine;
;;;
;;;        Kills any routine or game positioned after w on zecontrol.
;;;        It also deletes w from zecontrol.
;;;
define zakillafter(w);
vars l2;
rev(zecontrol)->zecontrol;
nil->l2;
while ispair(zecontrol) and
not(zatake(name)=hd(w) and zaentry(1)=hd(tl(w))) do
hd(zecontrol)::l2->l2;tl(zecontrol)->zecontrol;
endwhile;
l2->zecontrol;
enddefine;
```

```
*** AUXFUNS.p (3 of 4) ***
;;;
;;;         Finds first occurrence of goal x in zecontrol.
;;;
define zaworkthru(p);
vars l2 p1;p->p1;zecontrol->l2;rev(zecontrol)->zecontrol;
while ispair(zecontrol) do
while ispair(p1) do
if zaentry(1)=hd(p1)  and zatake(goaltype)=hd(tl(p1)) then
 return(zatake(name)::[^(zaentry(1))],l2->zecontrol) endif;
tl(tl(p1))->p1;
endwhile;
tl(zecontrol)->zecontrol;p->p1;
endwhile;
if zecontrol=[] then l2->zecontrol;return([none]);endif;
enddefine;
;;;
;;;         Works out list of empty entries for r/g w.
;;;
define zaents(w)->l;
nil->l;
valof(w)->w;
while ispair(w) do
if hd(hd(tl(w)))="*" then hd(w)::(nil::l)->l endif;
tl(tl(w))->w;
endwhile;
enddefine;


;;;
;;;         Used to move place in a routine or game.
;;;
define zagoto(n);
zaput(place,n);
enddefine;
;;;
;;;         Provides new numbers for use as indexes
;;;
vars zacount; 1->zacount;
define zaindex;
          zacount; zacount+1->zacount;
enddefine;
;;;
;;;         Compares two world situations and returns the factor on which they
;;;         differ, or [nothing] if they are the same.
;;;
define zadiff(l1,l2);
while ispair(l1) do
          if hd(tl(l1))/=suc2(l2,hd(l1))
          then
         if suc2(l2,hd(l1))=undef
         or suc2(l1,hd(l1))=undef
                    then [%undef,hd(l1)%]
                    else [%hd(l1)%]
                    endif; return
                    else tl(tl(tl(l1)))->l1;
          endif;
endwhile;[nothing]
enddefine;
;;;
;;;         Jumps to r/g x, killing procedures above x in control
;;;
define zajump(x);
while length(zecontrol)>1 and zatake(name)/=x
do tl(zecontrol)->zecontrol;
endwhile;
enddefine;
```

```
*** AUXFUNS.p (4 of 4) ***
;;;
;;;         Tests two association lists for equality.
;;;         (used for rules)
;;;
define zadiffs(l1,l2);
while ispair(l1) do
    if hd(tl(l1))/=suc(l2,hd(l1))
    then if suc(l2,hd(l1))=undef
        then [%undef,hd(l1)%]
        else [%hd(l1)%]
        endif; return
        else tl(tl(l1))->l1;
    endif;
endwhile;[nothing]
enddefine;
define zaequal(l1,l2);
zadiffs(l1,l2)=[nothing];
enddefine;
;;;
;;;         Makes a question to discover the position of x.(p298)
;;;

define zamakeq(x);
[is]<>[%x,zctprops,x.zctypof.suc.hd%];
enddefine;
;;;
;;;         Returns true if there is no game in progress and false otherwise.
;;;
define zanogame;
vars l;zecontrol->l;
while ispair(l)
do if suc(hd(l),kind)="game" then return(false);endif;tl(l)->l;
endwhile;null(zeplace);
enddefine;
;;;
;;;         Removes entries in structure x and begins at 1.
;;;
define zawipe(x);
zaput1(x,place,1);
zaput1(x,entries,zaents(x));
if zenogame() then return; endif;
if zatake1(x,colour)="black" then zaput1(x,expectreply,true)
else zaput1(x,expectreply,false);endif;
enddefine;
```

```
*** EXEC.p  (1 of 6) ***
;;;
;;;          (Power 1974,p299) Executive functions which interpret the routines and games
;;;
vars zecontrol,zebox,zeagain,zenext,zejoint,zegame1,zemove1,zehold,zeplace;
;;;
;;;          Exits from the current game or routine(p302)
;;;
define zeexit;
if pro or pro=1 then zosp(); sp(1); pr(zatake(name));
prs(' ended by '); pr(zkme) ; endif;
if zatake1("zrassess",name)="zrassess" or zatake1("zrachieve",name)="zrachieve"
  then
   if (zatake(name)="zgrule" or zatake(name)="zgask" or zatake(name)="zgtell")
    then if zaentry(1)/=[] and zaentry(2)/=[] then
         [%zatake(name),zaentry(1),zaentry(2),
          if zatake1("zrassess",name)="zrassess" then
             [^(zatake1("zrassess",goaltype))]
          else
             [^(zatake1("zrachieve",goaltype))]
          endif
         %]::zginformation
         ->zginformation;
       endif;
   endif;
else
endif;
tl(zecontrol) -> zecontrol;
enddefine;
;;;
;;;          Returns true if top procedure in control is not a game else false
;;;
define zenogame;
zatake(kind)="routine";
enddefine;
;;;
;;;          Returns true if place reached is marked and false if not
;;;
define zeatmark;
[%zatake(name),zatake(place)%]=zeplace;
enddefine;
;;;
;;;          Restores the interrupted game when a mark is reached.
;;;
define zerevert;
zehold::zecontrol->zecontrol;
nil->zeplace;
nil->zehold;
enddefine;
;;;
;;;          Interpreter for routines
;;;
define zerout();
vars c,n;
if zeatmark() or zeatmark() =1 then return(zerevert());endif;
zatake(place)->n;
suc(valof(zatake(name)),n)->c;
if hd(c)="*" and (zaentry(n)/=nil)
then zaput(place,n+1);
else zobug(); popval([%".",hd(tl(c))%]);
endif;
enddefine;
```

```
*** EXEC.p (2 of 6) ***
;;;
;;;         Translates message from english into an entry, using the list of
;;;         functions t obtained from the game definitions.(p304)
;;;
vars oldmessage, oldtestfns;
define zeread(t);
vars m,e;
wmessage -> oldmessage;  t -> oldtestfns;
wmessage->m;
nil->wmessage;
if null(m) then return(nil) ;endif;

while ispair(t) do
        popval([%m,".",hd(t)%])->e;
        if e /= undef then return(e);
        else  tl(t)->t; endif;
endwhile;
pr('zeread: message: '); pr(oldmessage);
pr('   failed tests: '); npr(oldtestfns);
return([inapt]);        ;;;if null(t)
enddefine;
;;;
;;;         Loads game or routine w into zecontrol
;;;
define zeload(w);
if pro or pro=1 then nl(0);sp(1+(length(zecontrol)*2));pr(w);
prs(' loaded by ');pr(zkme);endif;
if w="zggame" then zakill(w); endif;
if zcgame(w) or zcgame(w)=1 then goto xgame endif;
xroutine:
rep(zcfshell,name,w) :: zecontrol->zecontrol;
zaput(place,1);
zaput(entries,zaents(w));
return;
xgame:
rep(zcgshell,name,w)::zecontrol->zecontrol;
zaput(place,1);
zaput(colour,white);
zaput(entries,zaents(w));
zaput(expectreply,false);
enddefine;
;;;
;;;         Calls the other robot by name
;;;
define zecall;
zepost([%zkyou%]);
zeload("zggame");
zaput(expectreply,true);
enddefine;
```

```
*** EXEC.p (3 of 6) ***
;;;
;;;         Arranges for game g to be played with first move m, calling partner
;;;         if necessary in order to get zggame loaded.
;;;
define zeplay(g,m);
[%g%]->zegame1;m->zemove1;
if zegame1=[zgask]  and hd(zemove1)="is" then
if zkinfoasked(tl(zemove1))=[asked] then .popready;npr('*1*');return(.zeexit);endif;
endif;
if zegame1=[zgask] and hd(zemove1)="can" then
if zkinfoasked(zemove1)=[asked] then npr('*2*');return;endif;
endif;
if zegame1=[zgtell] and tl(zemove1)/=[] and hd(tl(zemove1))="is" then
if zkinfoasked(tl(tl(zemove1)))=[asked] then npr('*3*');return;endif;
endif;
if zegame1=[zgtell] and (tl(zemove1)=[] or tl(tl(zemove1))=[]) then
if zkinfoasked(zemove1)=[asked] then npr('*4*');return;endif;
endif;
if zegame1=[zgrule] then
if zkinfoasked(zemove1)=[asked] then npr('*5*');return;endif;
endif;
if (zejoint or zejoint=1) and (zanogame() or zanogame()=1) and g/="jump"
then zeload("zggame");[.zecont]->zenext;
else .zecall;endif;
if ispair(zehold) and suc(zehold,name)="zggame"then nil->zehold;nil->zeplace;endif;
enddefine;
;;;
;;;         Interpreter for games
;;;
define zegame();
vars c,n,mine,entry,made,message,tests;
begin:
zatake(place)->n;
suc(valof(zatake(name)),n)->c;
if c=undef then zeexit() endif;
(hd(c)="*")->entry;
if entry or entry=1 then (ispair(zaentry(n)))->made endif;
(hd(tl(c))=zatake(colour))->mine;
if mine or mine=1
then if entry or entry=1
        then if made or made=1 then goto advance else goto perform endif;
        else goto perform
        endif;
else if entry or entry=1
        then if made or made=1  then goto advance else goto read endif;
        else goto advance
        endif;
endif;
perform:
zobug();
popval([%".",hd(tl(tl(c)))%]); return;
advance:
zaput(place,n+1); return;
read:
rev(tl(rev(tl(tl(tl(tl(c)))))))->tests;
zeread(tests)->message;
if null(message)  then goto swap;
elseif message=[inapt] then goto moan ;
else return( if zctgame()>0 then zaput(expectreply,false) endif,zaenter(message));
endif;
moan:
pr('  moaning...  message = '); npr(message);zeplay("jump",undef); return;
swap:
zepost([--]);
enddefine;
```

```
*** EXEC.p (4 of 6) ***
;;;
;;;        Returns true if robot has been called by partner and false if not.
;;;
define zecalled;
wmessage=[%zkme%];
enddefine;
;;;
;;;        Loads zggame and takes black
;;;
define zealert;
zeload("zggame");
zaput(colour,black);
enddefine;
;;;
;;;        Posts message m;
;;;
define zepost(m);
m->zebox;
false ->zeagain;
 zebox->wmessage;
 undef->zebox;
 [.zecont]->zenext;
enddefine;
;;;
;;;        Responds to being called.
;;;
define zeready;
zealert();zepost([yes]);
enddefine;
;;;
;;;        Used to initialise the ze variables before a run.
;;;
define zeprep;
nil->zecontrol;[]->zehold;false->zunexpected_event;
zeload("zrbasic"); false->zejoint;
[.zecont]->zenext; nil->zeplace; true->zeagain;
enddefine;
;;;
;;;         Gives control to structure below current one and marks it at place n
;;;
define zemark(p);
p->zeplace;hd(zecontrol)->zehold;tl(zecontrol)->zecontrol;
enddefine;
;;;
;;;        Jumps back to zrbasic 7. used when an inappropriate remark has been
;;;         made.
;;;
define zejump;
zajump("zrbasic");zagoto(7);
enddefine;
;;;
;;;         master function used by chairman to arouse robot
;;;
;;;
;;;        Runs either zecall,zesend or zecont depending on which was
;;;         last put into zenext.
;;;
define zeexec;
if (zenogame()) then zerout(); else zegame();endif;
enddefine;
```

```
*** EXEC.p (5 of 6) ***
;;;
;;;        Main arousal mechanism called by chairman.
;;;        The robot reacts according to the value of expectreply
;;;        on the control stack. The main choices are:
;;;
;;;        (1) A call by name.
;;;        (2) There is a message and we wanted it.
;;;        (3) There is a message and we didn't want it.
;;;        (4) There is no message and we want one.
;;;        (5) There is no message and we don't want one.
;;;
define zearouse;
if zecalled() or zecalled()=1 then            ;;;  call by name.
 if member(9,ppro) then pr(mt+1);pr(' ');
 pr('*9*');pr(zkme);pr(' is responding to being called');nl(1);
 endif;if zatake(expectreply)=false and zatake(place)/=1 then
        repeat forever
         if member(5,ppro) then pr(mt+1);pr(' ');
  pr('*5* ');pr(zkme);pr(' is finishing off an old conversational procedure');
      nl(1);
         endif;
        zegame();quitunless(zatake(expectreply)=false);
        endrepeat;endif;
return(zeready());
else
if zunexpected_event then
    until
     not(zatake(kind)="game" or zatake(name)="zrplan"
     or zatake(name)="zrfindout") do .zeexit;
    enduntil;nil->wmessage;
    false->zunexpected_event;zeload("zrassess");
    zaput(goaltype,"none");.zerout;return;
else
 if wmessage=nil or wmessage=[--] then
  if zctgame()>0 then        ;;; no message but we want one.
    if member(1,ppro) then pr(mt+1);pr(' ');
    pr('*1* ');pr(zkme);pr(' is waiting for an utterance ');nl(1);
    endif;
    if wmessage=[--] then    ;;; a message that's not worth reading
    nil->wmessage;if zenogame() then zerout()
            else zaput(place,zatake(place)+2);zegame();endif;
    else return(false->zeagain);
    endif;
  else                  ;;; no message and we don't want one.
    if member(2,ppro) then pr(mt+1);pr(' ');
    pr('*2* ');pr(zkme);pr(' is  planning ');nl(1);
    endif;
    if wmessage=[--] and member(1,ppro) then
    pr('*1* ');pr(zkme);pr(' is waiting for a goal or an action');nl(1);
    endif;
    nil->wmessage;
  endif;
 else
```

```
*** EXEC.p (6 of 6) ***
  if zctgame()>0 then         ;;; there is a message and we want it.
    if zenogame() then repeat forever zerout();quitunless(zenogame());
    endrepeat;else zegame();endif;
    if member(3,ppro) then pr(mt+1);pr(' ');
    pr('*3* expected utterance for ');pr(zkme);nl(1);
    endif;
  else                    ;;; a message and we weren't expecting it.
    if member(4,ppro) then pr(mt+1);pr(' ');
    pr('*4* unexpected utterance for ');pr(zkme);nl(1);
    endif;
     if zenogame() and zatake(place)/=8
     then repeat forever zerout();quitunless(zenogame() and zatake(place)/=8);
     endrepeat;
      if zatake(expectreply)=true then return(zegame());endif;
     endif;
           if zatake(expectreply)=false and zatake(place)/=1 then
            repeat forever
             if member(5,ppro) then pr(mt+1);pr(' ');
      pr('*5* ');pr(zkme);pr(' is finishing off an old conversational procedure');
      nl(1);
             endif;
            zegame();quitunless(zatake(expectreply)=false);
            endrepeat;
            if zatake(expectreply)/=true then zealert();
             if member(6,ppro) then
             pr(mt+1);pr(' ');
             pr('*6* ');pr(zkme);pr(' has no other conversational procedures to contend with');nl(1);
             endif;
            else
      if member(7,ppro) then
      pr(mt+1);pr(' ');
      pr('*7* ');pr(zkme);pr(' has other conversational procedures to contend with');
      nl(1);
      endif;
            endif;
           else
            if member(9,ppro) then
            pr(mt+1);pr(' ');
     pr('*9* ');pr(zkme);pr(' has no conversational procedures to contend with');
     nl(1);
     endif;
           zealert();
    endif;
  endif;
 endif;
endif;
endif;
zeexec();
enddefine;
```

```
*** WRITE.p (1 of 3) ***
;;;
;;;          These functions translate game entries into english in an
;;;           unprincipled manner.(Power 1974, p305)
;;;

;;;  zggame 1 and 2;
define zwname x;
[zgask    [may I ask you something]
 zgtell   [I want to tell you something]
 zgrule   [I want to explain something]
 zggoal   [I want to suggest a goal]
 zgassess [%if zkme=suc(zpactor,zpcurr)  then
  "lets","assess","the","result","of","my","action"
 else "lets","assess","the","result","of","your","action" endif;%]
 zgplan   [shall we make a plan]
 zgexperiment [shall we do an experiment]
 zgtest [shall we do a test]
 jump      [we have got muddled; lets start again]],
 hd(x).suc;
 enddefine;
 ;;;
 define zwready x;
 if memb(hd(zaentry(1)),[zgplan zgcheck zgassess jump])
 then [ok] else [go ahead] endif;
 enddefine;
 ;;;
 ;;;          zgask 1 and 2
 ;;;
define zwquery(x);
 vars v,g;
 hd(x)->v;tl(x)->g;
;;; the elseif 4 lines down doesn't work when Mary's goal is to open
;;; the door. - cb.
 if hd(g)=zkme then
            [I]<>tl(g)->g;
            if v="is" then "am"->v endif;endif;
 if hd(g)=zkyou then
            [you]<>tl(g)->g;
            if v="is" then "are"->v endif;
 endif;
 if memb(hd(g),[door bolt]) then [the]<>g->g endif;
 if hd(tl(g))=push then g<>[the door]->g
 elseif hd(tl(g))=slide then g<>[the bolt]->g endif;
 v::g;
enddefine;


 ;;;
 define zwanswer(x);
 if x=[undef] then [I dont know] else x endif;
  enddefine;
 ;;;
 ;;;  zggoal 1 and 2
 ;;;
 define zwplead(x);
 if hd(x)=zkme then tl(x)
 elseif hd(x)=zkyou then [you]<>tl(x)
 else [the]<>x
 endif;->x;
 [will you help me get]<>x;
 enddefine;
 ;;;
 define zwreact(x);
 if x=[yes] then [by all means] else [no]; endif;
 enddefine;
```

```
*** WRITE.p (2 of 3) ***
;;;
;;;         zgtell 1 and 2
;;;
define zwrelate(x);
if zcevent(x) then if hd(x)=zkme then return(suc(
                    [push [I have pushed the door] slide [I have slid the bolt] move [I have moved]], hd(tl(x)) ));
          endif;
          if hd(x)=zkyou then return(suc(
          [push [you have pushed the door]
           slide [you have slid the bolt]
            move [you have moved]],
          hd(tl(x)) ));
         else return(suc(
          [push [somebody has pushed the door]
           slide [somebody has slid the bolt]
            move [somebody has been moved]],
          hd(tl(x)) ));
          endif;
 endif;
if zcobject(x)
then return(zwchange(x));endif;
 vars t; hd(x)->t;
 zwquery(tl(x))->x;
 if hd(tl(x))="the"
 then [the]<>[%hd(tl(tl(x))),hd(x)%]; tl(tl(tl(x)))->x
 else [%hd(tl(x)),hd(x)%];tl(tl(x))->x;
 endif;
 if t=0 or t=false then <>[not] endif;
 <>x;
enddefine;
;;;
define zwwreply(x);
 suc([yes [I already know that]
            no  [I disagree]
            undef [I see]],
  hd(x));
enddefine;
;;;
define zwexamine (x);
 zwwreply(x);
enddefine;
;;;
;;;         zgrule 1,2,3,4 and 5
;;;
 define zwwrule(x);
 vars e,s,r;
 suc(x,evt)->e;suc(x,sit)->s;suc(x,res)->r;
 [if]<>tl(zwquery("can"::(zkyou::tl(e))))->e;
 if s=[any]
 then nil
 elseif hd(s)/="robot" then [%"when","the",hd(s),"is",hd(tl(s))%]
 else [when you are]<>[%hd(tl(s))%]
 endif;->s;
 if r=[nothing] or r=[undef] then [, nothing happens]
 else if hd(r)="robot" then [, you change] else [, the]<>r
 <>[changes] endif;<>[position] endif;->r;e<>s<>r;
 enddefine;
 ;;;
 define zwbrule(x);
 zwwrule(x);
 enddefine;
 ;;;
 define zwwnote(x);
 [umm [I see]],x.hd.suc;
 enddefine;
```

*** WRITE.p (3 of 3) ***

```
 ;;;
 define zwbreply(x);
 zwwreply(x);
 enddefine;
 ;;;
 ;;;        zgplan 1 and 2
 ;;;
define zwsuggest(x);
 vars s;
 if x=[no] then return([ I cant think of one]) endif;
 zpsit(x)->s;
 if s/=undef
 then tl(zwquery([is]<>s))->s;if hd(s)="I" then
  "me"::tl(s)->s endif;
            [we get]<>s<>[and then]
 else nil
 endif;->s;
 zpevt(x)->x;
 tl(zwquery([can]<>x))->x;
 [I suggest that]<>s<>x;
enddefine;

 ;;;
 define zwrespond(x);
 vars p;zaentry(1)->p;
 if p=[no] then [yes [I will then] no [oh]]
 else [yes [all right] no [I disagree]]
 endif;hd(x).suc;
 enddefine;
 ;;;
;;;
 ;;;         zgtest 1 and 2
 ;;;
vars zwstart zwparticipate; zwsuggest->zwstart;zwrespond->zwparticipate;
 ;;;
 ;;;        zgexperiment 1 and 2
 ;;;
vars zwmake zwaccept;zwsuggest->zwmake;zwrespond->zwaccept;
 ;;;
 ;;;         zgassess 1,2,3 and 4
 ;;;
 define zwchange(x);
 if x=[nothing]  then return([nothing has happened])
 elseif hd(x)=zkme then [I have]
 elseif hd(x)=zkyou then [you have]
 else [the]<>x<>[has]
 endif;<>[changed position];
 enddefine;
 ;;;
 define zwconfirm(x);
 [yes [yes] no [I disagree]],hd(x).suc;
 enddefine;
 ;;;
 define zwparent(x);
 rev(zwrelate(x))->x;hd(x);
 if memb("not",x) then ::[yet] else ::[now] endif;
 <>tl(x)->x;rev(x);
 enddefine;
 ;;;
 define zwbenter(x);
 [right];
 enddefine;
```

```
*** DECIPHER.p (1 of 3) ***
;;;
;;;
;;;         The following functions are used to decode english utterances.
;;;
;;;
define zdhalf(l,x);
while hd(l)/=x do tl(l)->l endwhile;tl(l);
enddefine;
;;;
;;;         returns lists of the words before and after x in l
;;;
define zdsplit(l,x);
zdhalf(l,x);rev(zdhalf(rev(l),x));
enddefine;

;;;
;;;         Used mainly to replace variant forms of a word (e.g. am,are)
;;;         with a standard one is.
;;;
define zdclean (x);
vars l1,l2;nil->l1;
            ;;; replace variant word forms
x,"slid",slide.xch;  "pushed",push.xch;  "moved",move.xch->x;
x,"am","is".xch; "are","is".xch->x;
x,"Are","is".xch->x;

            ;;; remove 4 lead words from help sentences
if memb("help",x) and length(x)>4 then tl(tl(tl(tl(x))))->x endif;

            ;;; remove 2 lead words from suggest/want sentences
if length(x)>1 and memb(hd(tl(x)),[suggest want]) then tl(tl(x))->x endif;

            ;;; reorder clauses in slide/push sentences
if memb(slide,x) then zdsplit(x,slide)->l1->l2;l1<>[slide]->l1
elseif memb(push,x) then zdsplit(x,push)->l1->l2; l1<>[push]->l1 endif;

if ispair(l1) and length(l2)>1 then l1<>tl(tl(l2))->x endif;
;;;if ispair(l1) then
;;;         pr('zdclean:  '); pr(l1); sp(2); npr(l2);
;;;         if length(l2) > 1  then tl(tl(l2)) -> l2; endif;
;;;         l1 <> l2 -> x;
;;;endif;

x;
enddefine;
;;;
;;;         Replaces pronouns with their referents.
;;;
define zdprons (l);
l,"i",zkyou.xch; "me",zkyou.xch;"I",zkyou.xch; "you",zkme.xch;
 "we",both.xch;"somebody","robot".xch;
enddefine;
;;;
;;;         Given a list of words x and an english expression l,
;;;         returns the first word in x which is also in l, and
;;;         undef if none are.
;;;
define zdfind(l,x);
while ispair(x)
do if memb(hd(x),l) then hd(x); return else tl(x)->x endif;
endwhile; undef;
enddefine;
```

```
*** DECIPHER.p (2 of 3) ***
;;;
;;;          The following functions are used to decode different kinds of
;;;          utterances. if the english expression x can be construed as the
;;;          kind of entry wanted, the entry is returned; if not, undef
;;;          is returned. thus zdgame tries to interpret x as a game suggestion
;;;          zdquery tries to interpret it as a query, and so on.
;;;


;;;
define zdgame(x);
vars l;
[ask zgask goal zggoal next zgcheck explain zgrule tell zgtell assess
zgassess plan zgplan experiment zgexperiment test zgtest start jump]->l;
suc(l,zdfind(l,x))->l;
if l=undef then undef else l::nil endif;
enddefine;

define zdsitn(x);
          zdprons(zdclean(x))->x; x<>[%zkyou%]->x;
          if length(x)<2 then return(undef); endif;
          (zdfind(zcobjs,x))::[%zdfind(zcprops,x)%]->x;

          if memb(undef,x) or not(zcpropof(hd(tl(x)),zctypof(hd(x))))
          then undef else x endif;
enddefine;

define zdevent (x);
zdprons(zdclean(x))->x;
if length(x)<2 then return(undef); endif;
(zdfind([John Mary robot],x))::[%zdfind(zcacts,x)%]->x;
if memb(undef,x) then undef else x endif;
enddefine;

;;;
define zdquery (x);
vars a;
zdprons(zdclean(x))->x;
if length(x)<3 or not(memb(hd(x),[can is])) then return(undef) endif;
if hd(x)="can" then zdevent(tl(x)) else zdsitn(tl(x)) endif;->a;
if a=undef then undef else hd(x)::a endif;
enddefine;


;;;
;;;
define zdrule(x);
          vars e,s,r;
          xch(zdclean(x),"you","robot")->x;xch(x,"one","robot")->x;xch(x,"I","robot")->x;
          if hd(x)/="if" or not(memb(",",x)) then return(undef); endif;
          zdsplit(x,",")->e->r;
          if memb("when",e) then zdsplit(e,"when")->e->s else [any]->s endif;
          zdfind(zcacts,e)->x;
          if memb("robot",e) and x/=undef
                    then [robot]<>[%x%]->e
                    else return(undef);
          endif;
          if s/=[any]
          then (zdfind(zctypes,s))::[%zdfind(zcprops,s)%]->s;
                    if memb(undef,s) or not(zcpropof(hd(tl(s)),hd(s)))
                    then return(undef);
                    endif;
          endif;
          zdfind("nothing"::zctypes,r)::nil->r;
          if r=undef then return(undef) endif;
          [%evt,e,sit,s,res,r%];
enddefine;
```

```
*** DECIPHER.p (3 of 3) ***
;;;
;;;
define zdplan (x);
vars s,f;
zdprons(zdclean(x))->x;
if memb("and",x) then zdsplit(x,"and")->s->e
elseif memb("then",x) then zdsplit(x,"then")->s->e
else nil->s;x->e;
endif;
if ispair(s) then zdsitn(s)->s endif;
zdevent(e)->e;
if s=undef or e=undef then return(undef) endif;
if ispair(s) then both::[%s%]->s endif;
s<>[%hd(e),e%]
enddefine;

;;;
define zdsign(x);
vars y,n;
if x=[I see] then return([undef]) endif;
if memb("dont",x) and memb("know",x) then return([undef]) endif;
[know yes agree good can will right fine splendid ok go by]->y;
[no disagree cant wont bad oh lousy faulty silly]->n;
zdfind(n<>y,x)->x;
if x=undef then undef elseif memb(x,y) then [yes] else [no] endif;
enddefine;

;;;
define zdfact(x);
vars t;
zdclean(x)->x;
not(memb("not",x))->t;
zdquery(zdfind(x,[can is]) :: x) -> x;
if x=undef then undef else t::x endif;
enddefine;

;;;
define zdobj (x);
zdfind([nothing]<>zcobjs,zdprons(x)); ::nil;
enddefine;
```

```
*** PLAY.p (1 of 7) ***
;;;
;;;        (Power 1974, p317) various functions for starting the simulation in different ways
;;;
;;;
;;;
;;;        Prints out a robots utterances.
;;;
define write(l);
vars l;
if l=undef then return(prs('<undef>'));
 elseif null(l) then return(prs(' --'));
endif;
consstring(destword(hd(l)))->hd(l);
lowertoupper(subscrs(1,hd(l)))->
subscrs(1,hd(l));
if hd(l)='Lets' then  'Let\'s'->hd(l); endif;
pr(hd(l));
consword(deststring(hd(l)))->hd(l);
if memb(hd(l),[Shall Are Will May Can Is]) then [^^(l) ?]->l else
 [^^(l) .]->l;
endif;

if memb(hd(l),[Mary John I]) then else
 consstring(destword(hd(l)))->hd(l);
 uppertolower(subscrs(1,hd(l)))->
 subscrs(1,hd(l));
 consword(deststring(hd(l)))->hd(l);
 if wmessage/=nil then
   consstring(destword(hd(wmessage)))->hd(wmessage);
   uppertolower(subscrs(1,hd(wmessage)))->
   subscrs(1,hd(wmessage));
   consword(deststring(hd(wmessage)))->hd(wmessage);
 endif;
endif;
tl(l)->l;
while ispair(l)
do
if memb(hd(l),[cant]) then 'can\'t'->hd(l) endif;
if hd(l)="somethin" then prs(' something')
elseif hd(l)="," or hd(l)=";" or hd(l)="." or hd(l)="?" then pr(hd(l))
else sp(1); pr(hd(l)) endif;tl(l)->l;
endwhile;
enddefine;
vars zostart;
```

```
*** PLAY.p (2 of 7) ***
;;;
;;;           The chairman
;;;
define replay;
jawake();zostart();jasleep();mawake();zostart();masleep();
 if erase(count//2) > 0 then goto john else goto mary endif;
john:
if count>stop then return(nl(2)) endif;
true->jeagain;
if ispair(Unexpected_events) and count>hd(Unexpected_events)
then
if zcevent(hd(tl(Unexpected_events))) then zpdo(hd(tl(Unexpected_events)));
else
wsub(hd(hd(tl(Unexpected_events))),hd(tl(hd(tl(Unexpected_events)))));.prw;
endif;
tl(tl(Unexpected_events))->Unexpected_events;true->munexpected_event;true->junexpected_event;
endif;
jawake();
repeat forever
if zaentry(8)=[achieved] then
return(nl(1),npr('***** Plan successful *****'),npr(jtopics),npr(mtopics));endif;
if zptree/=nil then
if (suc(zpstate,hd(suc(zptree,0)))) = "achieved"  then
return(nl(1),npr('***** Plan successful *****'),npr(jtopics),npr(mtopics));endif;
if (suc(zpstate,hd(suc(zptree,0)))) = "failed" then
return(nl(1),npr('***** Plan unsuccessful *****'));endif;
endif;
if zatake(place)=1 and zatake(colour)="white"
and zatake(expectreply)/=true and member(zatake(name),
[zgask zgtest zgexperiment zggoal zgcheck zgrule  zgtell zgassess zgplan zggame])
and wmessage=nil then nl(1);endif;
   .zearouse;1+jt->jt; if jt=jspeed then 0->jt;false->zeagain;endif;
quitunless(zeagain or zeagain=1);
endrepeat;
jasleep();
if null(wmessage)then goto mary;endif;if pro then nl(1); endif;pr(count);
1+count->count;prs(' John:'); false->prw1;write(wmessage);nl(1); if pro then nl(1); endif;
mary:
if count>stop then return(nl(3))  endif;true->meagain;mawake();
repeat forever
if zaentry(8)=[achieved] then
return(nl(1),npr('***** Plan successful *****'),npr(jtopics),npr(mtopics));endif;
if zptree/=nil then
if (suc(zpstate,hd(suc(zptree,0)))) = "achieved"  then
return(nl(1),npr('***** Plan successful *****'),npr(jtopics),npr(mtopics));endif;
if (suc(zpstate,hd(suc(zptree,0)))) = "failed" then
return(nl(1),npr('***** Plan unsuccessful *****'));endif;
endif;
   if zatake(place)=1 and zatake(colour)="white"
and zatake(expectreply)/=true and  member(zatake(name),
[zgask zgtest zgexperiment zggoal zgcheck zgrule  zgtell zgassess zgplan zggame])
and wmessage=nil then nl(1);endif; .zearouse; 1+mt->mt;
   if mt=mspeed then 0->mt;false->zeagain;endif;
quitunless(zeagain or zeagain=1);endrepeat;
masleep();
if null(wmessage)then goto john; endif;if pro then nl(1); endif;
pr(count);1+count->count;prs(' Mary:');  false->prw1;write(wmessage);nl(1);if pro then nl(1) endif;
 goto john;enddefine;
```

```
*** PLAY.p (3 of 7) ***
;;;
;;;         Runs a conversation with the first initial setting.
;;;
define run1;
reset1();
.zeprep;zaenter1("zrbasic",4,[failed]);
zunexpected_event->munexpected_event;zunexpected_event->junexpected_event;
zecontrol->mecontrol;zecontrol->jecontrol;
zejoint->mejoint;zejoint->jejoint;zehold->jehold;zehold->mehold;
zenext->menext;zenext->jenext;
zeplace->meplace;zeplace->jeplace;
zeagain->meagain;zeagain->jeagain;
nl(3);.prw;replay();
enddefine;
define run2;
reset1();1->count; zaenter1("zrbasic",4,[failed]);
zecontrol->jecontrol;
nl(3);.prw;gstart();
John,John->jkme->mkyou;
Mary,Mary->mkme->jkyou;
[John in]->jkgoal; [none]->mkgoal;
[John out Mary in bolt down door shut]->wobjects;
nil->wmessage;
[ move ]->jkacts;[push slide move]->mkacts;
[[evt[robot push]sit[bolt up]res[door]]
 [evt[robot slide]sit[any]res[bolt]]
 [evt[robot move]sit[any]res[nothing]]]->jcrshell;
 [[evt[robot push]sit[any]res[nothing]]
 [evt[robot slide]sit[any]res[bolt]]
 [evt[robot move]sit[door open]res[robot]]]->mcrshell;
 [door John Mary]->jksee;
 [Mary]->mksee;
jcwshell->jkworld;zklook();
jcrshell->jkrules;
jcwshell->jkxsee;
nil->jkxgoal;
if zpstart/="Super" then nil->jkxacts;
nil->jkxrules;endif;
mcwshell->mkworld;zklook();
mcrshell->zkrules;
mcwshell->zkxsee;
nil->mkxgoal;
if zpstart/="Super" then nil->mkxacts;
nil->mkxrules;endif;
replay();
enddefine;

;;;
;;;         The following functions allow interaction with the computer only.
;;;
vars stdmsgs;
[ [ ok][Mary][I want to ask you something][I want to tell you something] [I want to explain something]
 [lets assess the result of my action] [shall we make a plan?]
 [are you in?] [can you move?][is the door open?][the door is shut]
 [if you push when the bolt is up, the door moves] [I want to suggest a goal]
 [go ahead] [will you help me get in?] [by all means] [yes]
 [no] [John] [the door is open] [you are out][I am now in] [I have moved]
 [I have pushed the door] [I suggest that we get the door open and then you move]
 [I suggest that I push the door] [nothing has happened] [the door has changed position][the door is now open]
 [is the door shut?][is the bolt up?] [is the bolt down?] [are you in?]
 [are you out?][will you help me get the door open?]
 [will you help me get the door shut?] [if you push when the bolt is down, nothing happens]
 [if you move , nothing happens] [I disagree] [I suggest that we get the door open and then I move]
 [If you move when the door is open , you change position] [I have changed position]
 [I see] [all right] [right][I disagree . if you move when the door is open , you change position .] ] -> stdmsgs;
```

```
*** PLAY.p (4 of 7) ***
define read;
vars x,l, n, i;
nil->l;  length(stdmsgs) -> n;

until (.itemread.dup->x, x="." or x="?") do
      if x="--" then return(nil);
      elseif l=nil and isinteger(x) then
           if x=0 then  for i from 1 to n do
                pr(i); pr(': ');
                applist(stdmsgs(i), sp(%1%) <> pr); nl(1);
                endfor;
           elseif x<=n then
                ;;; npr(stdmsgs(x));
                applist(stdmsgs(x), sp(%1%) <> pr); pr('.'); nl(1);
                return(stdmsgs(x));
           else  x::l->l;
           endif;
      else  x::l->l;
      endif;
enduntil;
rev(l);
enddefine;
;;;
;;;          The chairman
;;;
define play;
vars x;
     erase(count//2)->x; nl(1); .prw; nl(1);
computer:
     if count>stop then nl(2); return() endif;
     if x>0 then
          zearouse();
                        if (wmessage/=nil) or (zkgoal=[none])
                        then goto human else goto computer endif;
human:
if pro then nl(1) endif;
     nl(1);pr(count);prs(' ');pr(zkme);prs(':'); write(wmessage); count+1->count;
     else 1->x
     endif;
     nl(2);
     if count>stop then return() endif;
     pr(count); prs(' ');pr(zkyou); read()->wmessage; count+1->count;
     goto computer;
enddefine;
;;;
;;;          mgo assumes the computer is mary jgo assumes computer to be john.
;;;          One can type numbers (see Power's linguistics 17 for coding);
;;;          remembering to type run before typing mgo or jgo and then when
;;;          a physical action occurs: to get to popready <option .> followed by
;;;          the appropriate action .wpush,.wslide,wmove and to continue again
;;;          <return>
;;;
define mgo;
reset1();zecontrol->mecontrol;0->jspeed;0->mspeed;
zaenter1("zrbasic",4,[failed]);
1->count;mawake();true->zeagain;true->meagain;1->count;play();
enddefine;
;;;
define jgo;
reset1();zecontrol->jecontrol;0->mspeed;0->jspeed;
zaenter1("zrbasic",4,[failed]);
1->count;jawake();true->zeagain;true->jeagain;1->count;play();
enddefine;
```

```
*** PLAY.p (5 of 7) ***
;;;
;;;        The following functions are for use by the Super Robot.
;;;        New function to combine two sets of rules s & t into one better set
;;;        contained in scrshell.
;;;
define srules(s,t);
vars r1,r2;
while s/=nil do
hd(s)->r1;zkrule1(t,r1,evt.suc)->r2;
if zaequal(r1,r2) then [%r1%]<>scrshell->scrshell;
elseif zkbetter(r1,r2) then [%r1%]<>scrshell->scrshell;
else [%r2%]<>scrshell->scrshell;
endif;tl(s)->s;
endwhile;
enddefine;
;;;
;;;        This function is used to make sure that each robot knows what they can
;;;        do before the start of a run. zkprep is also modified.
;;;
define sacts();
vars a;
mkacts->a;nil->jkxacts;
while a/=nil do
sub("jkxacts",hd(a),true);tl(a)->a;
endwhile;
jkacts->a;nil->mkxacts;
while a/=nil do
sub("mkxacts",hd(a),true);tl(a)->a;
endwhile;
enddefine;
;;;
;;;        Circular function that checks the internal plan tree zpgoal
;;;        and sees whether the current goal is already in the plan tree.
;;;
define newcircular(t,g);
vars c;0->c;
while ispair(t) do
if g=hd(tl(t)) then c+1->c; endif;
tl(tl(t))->t;
endwhile;
if c>1 then return(true) else return(false);endif;
enddefine;
;;;
;;;        This function is used to print out what the super robot knows
;;;        about the state of the two ordinary robots minds.
;;;
define zsstart;
nl(2); prs('*** what ');pr(jkme);prs(' capabilities are ***');nl(2);prs('1.   Goal:');
if null(jkgoal) or jkgoal=[none] then prs(' None') else write(zoplead(jkgoal)) endif;
nl(2); prs('2. Range of actions:');nl(1); zcacts->x;
while ispair(x) do sp(3);pr(hd(x));prs(': ');
if member(hd(x),jkacts) then pr('Yes');else pr('No');endif;pr(','); tl(x)->x;
endwhile;
nl(3);nl(2); prs('*** what ');pr(mkme);prs(' capabilities are ***');nl(2);prs('1.   Goal:');
if null(mkgoal) or mkgoal=[none] then prs(' None') else write(zoplead(mkgoal)) endif;
nl(2); prs('2. Range of actions:');nl(1); zcacts->x;
while ispair(x) do sp(3);pr(hd(x));prs(': ');
if member(hd(x),mkacts) then pr('Yes');else pr('No');endif;pr(','); tl(x)->x;
endwhile;
nl(3);prs(' Consequences of events:');nl(1);zcrshell->x;
while ispair(x) do nl(1);sp(3);pr(hd(tl(suc(hd(x),evt))));prs(':');nl(1);sp(2);
if suc(hd(x),res)=[undef] then prs(' undef') else write(zwwrule(hd(x)))
endif; tl(x)->x;
endwhile;nl(3);prs(' ********   Super plan prediction **********');nl(3);
enddefine;
```

*** PLAY.p (6 of 7) ***

```
;;;
;;;         splay is simplified from replay so that it only prints the relevant
;;;         plan.
;;;
define splay;
[]->mptree;[]->jptree;
jawake();jasleep();mawake();masleep();zsstart();
 if erase(count//2) > 0 then goto john else goto mary endif;
john:
zpgoal,zpcurr.suc->g;
if newcircular(zpgoal,g)=true then return(pr('circular plan detetected...'));
endif;
if zptree/=nil then
if (suc(zpstate,hd(suc(zptree,0)))) = "achieved" then return(nl(1),npr('***** Plan successful *****'));endif;
endif;
if count>stop then return(nl(2)) endif;
true->jeagain;
jawake();

repeat forever
   .zearouse;
            1+jt->jt;
   if jt=jspeed then 0->jt;false->zeagain;endif;
quitunless(zeagain or zeagain=1);
endrepeat;
jasleep();
if null(wmessage)
then goto mary;
endif;if pro then nl(1); endif;
;;;pr(count);
;;;         1+count->count;prs(' John:');
 false->prw1;
if zatake(name)="zgplan" and zatake(place)=1 and zatake(colour)="white" then
 nl(1);npr('Robot John suggests the following plan :');
npr(zaentry(1)); write(wmessage);nl(1);endif; if pro then nl(1); endif;
mary:
if count>stop then return(nl(3))  endif;
true->meagain;
mawake();
repeat forever
   .zearouse;1+mt->mt;
   if mt=mspeed then 0->mt;false->zeagain;endif;
quitunless(zeagain or zeagain=1);
endrepeat;
masleep();
if null(wmessage)
then goto john; endif;if pro then nl(1); endif;
 false->prw1;
if zatake(name)="zgplan" and zatake(place)=1 and zatake(colour)="white"then nl(1);
npr('Robot Mary suggests the following plan :');npr(zaentry(1));write(wmessage)
;nl(1);endif;if pro then nl(1) endif;
 goto john;
enddefine;
;;;
;;;         smerge combines goals, what it can see, what the ordinary robots believe
;;;         and what they can do.
;;;
define smerge();
[John Mary door bolt]->sksee;
if mkgoal=[none] then jkgoal->skgoal;jkgoal->jkxgoal else mkgoal->skgoal;
mkgoal->mkxgoal;endif;
[]->scrshell;
srules(jcrshell,mcrshell);
sacts();
enddefine;
```

```
*** PLAY.p (7 of 7) ***
;;;         Perfect knowledge is passed back to each robot and a simulation then
;;;         follows.
;;;
define resetsr;
nl(3);
npr(' ---------------------- Super Robot ----------------------------');
"Super"->zpstart;
smerge();
scrshell->mcrshell;
scrshell->jcrshell;
sksee->jksee;
sksee->mksee;
zecontrol->mecontrol;zecontrol->jecontrol;
zejoint->mejoint;zejoint->jejoint;
zenext->menext;zenext->jenext;
zeplace->meplace;zeplace->jeplace;
zeagain->meagain;zeagain->jeagain;
enddefine;

;;;
define sgo();
reset1();resetsr();1->count;
zaenter1("zrbasic",4,[failed]);
zecontrol->jecontrol;
nl(3);.prw;splay();
enddefine;
```

```
*** PRINT.p (1 of 2) ***
;;;
;;;          Various functions used to print  out the state of the program (e.g.
;;;          zoknow prints out a robots world model). (Power 1974, p317-318)
;;;


;;;
define zoplead (x);
tl(tl(zwplead(x)))->x;
if length(x)=3 then [%hd(x),hd(tl(x)),zkme,hd(tl(tl(x)))%] else x endif;
enddefine;


;;;
;;;          Prints states of programs
;;;
define  zosp;
nl(1);sp(length(tl(zecontrol))*2);
enddefine;


;;;
define zobug;
if pro then zosp(); pr(zatake(place));prs(' called by ');pr(zkme); endif;
enddefine;


;;;
define zoyn(x);
if x=1 or x=true then "yes" elseif x=0 or x=false then "no" else undef endif;
enddefine;
;;;
define zoknow();
vars x,y;
nl(2); prs('*** what ');pr(zkme);prs(' knows ***');
nl(2); prs('a.  world');nl(2);prs('    1. position of objects');
nl(1); zkworld->x;
while ispair(x) do nl(1); sp(3);pr(hd(x));prs(': ');pr(hd(tl(x)));
prs(': ');pr(hd(tl(tl(x))));tl(tl(tl(x)))->x;
endwhile;
nl(2);prs('    2.consequences of events');nl(1);zkrules->x;
while ispair(x) do nl(1);sp(3);pr(hd(tl(suc(hd(x),evt))));prs(':');nl(1);
sp(2);
if suc(hd(x),res)=[undef] then prs(' undef') else write(zwwrule(hd(x)))
endif; tl(x)->x;
endwhile;
nl(2); prs('b.   ');pr(zkyou);nl(2);prs('    1.  goal:');
if null(zkxgoal) then prs(' undef') else write(zoplead(zkxgoal)) endif;
nl(2); prs('    2. range of actions');nl(1); zcacts->x;
while ispair(x) do nl(1);sp(3);pr(hd(x));prs(': ');
pr(zoyn(suc(zkxacts,hd(x)))); tl(x)->x;
endwhile;
nl(2);prs('    3. knowledge of object positions');nl(1);
zkxsee->x;
while ispair(x) do nl(1); sp(3); pr(hd(x)); prs(':  ');
pr(zoyn(hd(tl(x))));tl(tl(x))->x;
endwhile;
nl(2);prs('    4. beliefs about consequences of events');
nl(1);zkxrules->x;nil->y;
while ispair(x)
do nl(1);sp(3);hd(tl(suc(hd(x),evt)))::y->y; pr(hd(y));prs(':');
nl(1);sp(2);write(zwwrule(hd(x)));tl(x)->x;
endwhile;zcacts->x;
while ispair(x)
do if memb(hd(x),y) then else nl(1);sp(3);pr(hd(x));prs(':');nl(1);
sp(3);pr(undef); endif;tl(x)->x;
endwhile;nl(2);prs('***********************************');nl(2);
enddefine;
```

```
*** PRINT .p (2 of 2) ***
;;;
define  zoplan;
prs( '*** ');pr(zkme);prs('s plan ***');
nl(1);prs('goals involved');pra(zpgoal);
nl(1);prs('subgoals of each goal');pra(zptree);
nl(1);prs('actors responsible');pra(zpactor);
nl(1);prs('state of each goal');pra(zpstate);
nl(1);prs('**********************************');nl(1);
enddefine;
;;;
define zocont;
vars x;
prs('*** ');pr(zkme);prs('s control structure ***');
rev(zecontrol)->x;
while ispair(x) do prc(hd(x));tl(x)->x; endwhile;
prs('**********************************');nl(1);
enddefine;
;;;
define zomind;
nl(4);
prs('*** current state of ');pr(zkme);prs('s mind ***');
zoknow();zoplan();zocont();
enddefine;
;;;
define zostart;
nl(2); prs('*** what ');pr(zkme);prs(' capabilities are ***');
nl(2);prs('1.   Goal:');
if null(zkgoal) or zkgoal=[none] then prs(' None') else write(zoplead(zkgoal)) endif;
nl(2);prs('2. Position of objects:');
nl(1);.zkprep; .zklook;zkworld->x;
while ispair(x) do  sp(3);pr(hd(x));prs(': ');pr(hd(tl(x)));prs(': ');
pr(hd(tl(tl(x))));pr(',');
tl(tl(tl(x)))->x;
endwhile;
nl(2);prs('3. Consequences of events:');nl(1);zcrshell->x;
while ispair(x) do nl(1);sp(3);pr(hd(tl(suc(hd(x),evt))));prs(':');nl(1);
sp(2);
if suc(hd(x),res)=[undef] then prs(' undef') else write(zwwrule(hd(x)))
endif; tl(x)->x;
endwhile;
nl(2); prs('4. Range of actions:');nl(1); zcacts->x;
while ispair(x) do sp(3);pr(hd(x));prs(': ');
if member(hd(x),zkacts) then pr('Yes');else pr('No');endif;pr(','); tl(x)->x;
endwhile;
nl(3);
enddefine;
```

Appendix 6

The following modifications perform the Monte Carlo simulation.

```
*** exhaustivetest .p (1 of 3) ***
;;;
;;;         The following modifications perform the Monte_Carlo simulation.
;;;          To execute type:
;;;          jglbal();
;;;
vars glbalacts,glbalsee,glbalspeed,glbalrule,files;
[
'A1.dat' 'A2.dat' 'A3.dat' 'A4.dat' 'A5.dat' 'A6.dat'
'A7.dat' 'A8.dat' 'A9.dat' 'A10.dat' 'A11.dat'
'A12.dat' 'A13.dat' 'A14.dat'
'A15.dat' 'A16.dat' 'A17.dat' 'A18.dat'
'A19.dat' 'A20.dat' 'A21.dat' 'A22.dat' 'A23.dat' 'A24.dat'
'A25.dat' 'A26.dat' 'A27.dat' 'A28.dat'
'A29.dat' 'A30.dat' 'A31.dat' 'A32.dat' 'A33.dat' 'A34.dat'
'A35.dat' 'A36.dat' 'A37.dat' 'A38.dat'
'A39.dat' 'A40.dat' 'A41.dat' 'A42.dat' 'A43.dat' 'A44.dat'
'A45.dat' 'A46.dat' 'A47.dat' 'A48.dat'
'A49.dat' 'A50.dat' 'A51.dat' 'A52.dat' 'A53.dat' 'A54.dat'
'A55.dat']->files;
vars cr lf;
cons_with consstring {% 13 %} -> cr;
cons_with consstring {% 10 %} -> lf;
lib datafile;
lib filelistsin;
[[move push slide][move]]->glbalacts;
[[door bolt][]]->glbalsee;
[3 1]->glbalspeed;
[
  [
   [evt[robot slide]sit[any]res[undef]]
   [evt[robot push]sit[bolt up]res[door]]
   [evt[robot move]sit[door open]res[robot]]
  ]
  [
   [evt[robot slide]sit[any]res[undef]]
   [evt[robot push]sit[any]res[undef]]
   [evt[robot move]sit[door open]res[robot]]
  ]
]->glbalrule;

define jglbal;
vars xx,ww,yy,zz,x1,w1,y1,z1,cnt,filecnt,tmpout,tmp2out,writeout,write2out,
dd,uttlist,utterancecnt,actioncnt,interruptioncnt,waitcnt,mcpcnt,jcpcnt,mskills,jskills;
npr('here');
0->cnt;0->filecnt;[]->tmpout;[]->tmp2out;[]->writeout;[]->write2out;
[]->jkacts;[]->jksee;[]->jspeed;[]->jcrshell;
for xx in glbalacts do;
xx->jkacts;
for yy in glbalsee do;[John ^^yy]->jksee;
for ww in glbalrule do;ww->jcrshell;
for zz in glbalspeed do;zz->jspeed;
    []->mkacts;[]->mksee;[]->mspeed;[]->mcrshell;
    for x1 in glbalacts do;x1->mkacts;
    for y1 in glbalsee do;[Mary ^^y1]->mksee;
    for w1 in glbalrule do;w1->mcrshell;
    for z1 in glbalspeed do; z1->mspeed;
cnt+1->cnt;if cnt < 1 then goto skip endif;
npr(cnt);filecnt+1->filecnt;
if filecnt=5 then npr('...');write2out->datafile(popfolder<>'Power(V.5.9):'<>hd(files));
pr('Writing ');pr(cnt-filecnt);pr('-');pr(cnt-1);pr(' recs to file ');npr(popfolder<>'Power(V.5.9):'<>hd(files));
tl(files)->files;[]->writeout;[]->write2out;0->filecnt;
endif;
0->jskills;0->mskills;
if suc(hd(tl(jcrshell)),res)= [undef] then jskills->jskills else 1+jskills->jskills endif;
if suc(hd(tl(mcrshell)),res)= [undef] then  mskills->mskills else 1+mskills->mskills endif;
if length(jkacts)= 1 then  jskills->jskills else 1+jskills->jskills endif;
if length(mkacts)= 1 then  mskills->mskills else 1+mskills->mskills endif;
if length(jksee)= 1 then jskills->jskills else 1+jskills->jskills endif;
if length(mksee)= 1 then  mskills->mskills else 1+mskills->mskills endif;
[^cnt ^(if suc(hd(tl(jcrshell)),res)= [undef] then "UNK" else "KNW" endif)
     ^(if suc(hd(tl(mcrshell)),res)= [undef] then "UNK" else "KNW" endif)
     ^(if length(jkacts)= 1 then "IMPRACT" else "PRACT" endif)
     ^(if length(mkacts)= 1 then "IMPRACT"else "PRACT" endif)
     ^(if length(jksee)= 1 then "IMPERC" else "PERC" endif)
     ^(if length(mksee)= 1 then "IMPERC" else "PERC" endif)
     ^(if jspeed= 1 then "SLOW" else "FAST" endif)
     ^(if mspeed= 1 then "SLOW" else "FAST" endif) ^jskills ^mskills
]->tmp2out;
          reset2();
skip:
     endfor;
     endfor;
     endfor;
     endfor;
endfor;
endfor;
endfor;
endfor;
nl(1);pr('Total number of dialogues = ');npr(cnt);
write2out->datafile(popfolder<>'Power(V.5.9):'<>hd(files));
pr('Writing ');pr(cnt-filecnt);pr('-');pr(cnt-1);pr(' recs to file ');npr(popfolder<>'Power(V.5.9):'<>hd(files));
tl(files)->files;[]->writeout;[]->write2out;0->filecnt;0->cnt;
enddefine;
```

```
*** exhaustivetest .p (2 of 3) ***
define reset2;
John,John->jkme->mkyou;
Mary,Mary->mkme->jkyou;
[door open]->jkgoal; [none]->mkgoal;
[John out Mary in bolt up door shut]->wobjects;
nil->wmessage;
 1->count;300->stop;false->prw1;false->pro;
false->ppro;0->mt; 0->jt;
 []->mptree;[]->mpstate;[]->mpgoal;[]->mpcurr;
[]->Unexpected_events;
 []->jptree;[]->jpstate;[]->jpgoal;[]->jpcurr;
 []->zptree;[]->zpstate;[]->zpgoal;[]->zpcurr;
 []->zecontrol;
 1->macount;1->jacount;
  if ppro=true then [1 2 3 4 5 6 7 8 9 10 11 12 13]->ppro;else []->ppro;endif;
zunexpected_event->munexpected_event;zunexpected_event->junexpected_event;
zecontrol->mecontrol;zecontrol->jecontrol;
zejoint->mejoint;zejoint->jejoint;zehold->jehold;zehold->mehold;
zenext->menext;zenext->jenext;
zeplace->meplace;zeplace->jeplace;
zeagain->meagain;zeagain->jeagain;
0->waitcnt;0->utterancecnt;0->interruptioncnt;0->actioncnt;[[x x x]]->uttlist;
0->mcpcnt;0->jcpcnt;
replay2();[^^tmp2out ^(if not(member(last(tmp2out),[CIRCULAR SUCCESSFUL UNSUCCESSFUL])) then
 "INCOMPLETE" endif) ^utterancecnt ^interruptioncnt ^actioncnt ^waitcnt
^jcpcnt ^mcpcnt ^(rev(uttlist))]->tmp2out;[^^write2out ^tmp2out]->write2out;
enddefine;
define replay2;
vars g;
jawake();.zeprep;.zrmaingl;.zrprep;.zrprep1;zaenter1("zrbasic",4,[failed]);;
jasleep();mawake();.zeprep;.zrmaingl;.zrprep;.zrprep1;zaenter1("zrbasic",4,[failed]);
masleep();
 if erase(count//2) > 0 then goto john else goto mary endif;
john:
zpgoaltype,zpcurr.suc->gt;zpgoal,zpcurr.suc->g;
if newcircular(zpgoal,zpgoaltype,g,gt)=true then return(
tmp2out<>[CIRCULAR]->tmp2out,
pr('circular plan detetected...Dialogue '),npr(cnt));
endif;
if count>stop then return(nl(2)) endif;true->jeagain;
if ispair(Unexpected_events) and count>hd(Unexpected_events)
then  if zcevent(hd(tl(Unexpected_events))) then zpdo(hd(tl(Unexpected_events)));
else wsub(hd(hd(tl(Unexpected_events))),hd(tl(hd(tl(Unexpected_events)))));.prw;
endif;
tl(tl(Unexpected_events))->Unexpected_events;true->munexpected_event;true->junexpected_event;
endif;
jawake();
repeat forever
if zaentry(8)=[achieved] then return(tmp2out<>[SUCCESSFUL]->tmp2out);endif;
if zptree/=nil then
if (suc(zpstate,hd(suc(zptree,0)))) = "achieved"  then return(tmp2out<>[SUCCESSFUL]->tmp2out);endif;
if (suc(zpstate,hd(suc(zptree,0)))) = "failed" then return(tmp2out<>[UNSUCCESSFUL]->tmp2out);endif;
endif;
    .zearouse;
               1+jt->jt;
    if jt=jspeed then 0->jt;false->zeagain;endif;
quitunless(zeagain or zeagain=1);
endrepeat;
jasleep();
if null(wmessage)
then goto mary;
endif;
             if hd(wmessage)="--" then waitcnt+1->waitcnt endif;
             1+count->count;utterancecnt+1->utterancecnt;
             if hd(tl(hd(uttlist)))='John' then interruptioncnt+1->interruptioncnt;endif;
             [%(count-1),'John',' :'%]<>wmessage->dd;
             [%dd%]<>uttlist->uttlist;
 false->prw1;
mary:
zpgoaltype,zpcurr.suc->gt;zpgoal,zpcurr.suc->g;
if newcircular(zpgoal,zpgoaltype,g,gt)=true then return(tmp2out<>[CIRCULAR]->tmp2out,
pr('circular plan detetected...Dialogue '),npr(cnt));
endif;
if count>stop then return(nl(3))  endif;true->meagain;mawake();
repeat forever
if zaentry(8)=[achieved] then return(tmp2out<>[SUCCESSFUL]->tmp2out);endif;
if zptree/=nil then
if (suc(zpstate,hd(suc(zptree,0)))) = "achieved"  then return(tmp2out<>[SUCCESSFUL]->tmp2out);endif;
if (suc(zpstate,hd(suc(zptree,0)))) = "failed" then return(tmp2out<>[UNSUCCESSFUL]->tmp2out);endif;
endif;
    .zearouse;1+mt->mt; if mt=mspeed then 0->mt;false->zeagain;endif;
quitunless(zeagain or zeagain=1);
endrepeat;
masleep();
if null(wmessage)
then goto john; endif;
            if hd(wmessage)="--"then waitcnt+1->waitcnt endif;
                         1+count->count;utterancecnt+1->utterancecnt;
               if hd(tl(hd(uttlist)))='Mary' then
  interruptioncnt+1->interruptioncnt;endif;[%(count-1),'Mary',' :'%]<>wmessage->dd;[%dd%]<>uttlist->uttlist;
 false->prw1; goto john;
enddefine;
```

```
*** exhaustivetest .p (3 of 3) ***
define zpdo (e);
actioncnt+1->actioncnt;
hd(e);hd(tl(e))->e;
if e=push then wpush()
elseif e=slide then wslide()
elseif e=move then wmove()
endif;
enddefine;

define zegame();
vars c,n,mine,entry,made,message,tests;
begin:
zatake(place)->n;
suc(valof(zatake(name)),n)->c;
if c=undef then zeexit() endif;
(hd(c)="*")->entry;
if entry or entry=1 then (ispair(zaentry(n)))->made endif;
(hd(tl(c))=zatake(colour))->mine;
if mine or mine=1
then if entry or entry=1
            then if made or made=1 then goto advance else goto perform endif;
            else goto perform
            endif;
else if entry or entry=1
            then if made or made=1  then goto advance else goto read endif;
            else goto advance
            endif;
endif;
perform:
zobug();
if n=1 and zatake(colour)=white then
if zkme="John" then jcpcnt+1->jcpcnt endif;
if zkme="Mary" then mcpcnt+1->mcpcnt endif;
endif;
popval([%".",hd(tl(tl(c)))%]); return;
advance:
zaput(place,n+1); return;
read:
rev(tl(rev(tl(tl(tl(tl(c)))))))->tests;
zeread(tests)->message;
if null(message)  then goto swap;
elseif message=[inapt] then goto moan ;
else return( if zctgame()>0 then zaput(expectreply,false) endif,zaenter(message));
endif;
moan:
pr('  moaning...  message = '); npr(message);
zeplay("jump",undef); return
swap:
zepost([--]);
enddefine;

define newcircular(t,tt,g,gt);
vars c;0->c;
while ispair(t) do
if g="undef" then [undef]->g endif;
if gt="undef" then [undef]->gt endif;
if ([%gt%]<>g=[%hd(tl(tt))%]<>hd(tl(t))) then c+1->c; endif;
tl(tl(t))->t;tl(tl(tt))->tt;
endwhile;
if c>1 then return(true) else return(false);endif;
enddefine;


define prw;
true->prw1;
;;;prs('(State of the world is now ');pr(wobjects);prs(')');
;;;if pro  then nl(1) endif;nl(1);
enddefine;
```