



UNIVERSITY *of*
ROCHESTER

Computer Science Area Paper

**MANAGEMENT OF OPEN-DOMAIN
DIALOGUE FOR VIRTUAL
CONVERSATIONAL AGENTS**

Benjamin Kane

Area Committee

Dr. Lenhart Schubert

Dr. Aaron Steven White

Dr. James Allen

April 2021

University of Rochester
Department of Computer Science

Abstract

The development of “intelligent” dialogue assistants and conversational agents, capable of being transferred from one domain to another with minimal engineering necessary, has been consistently high on the priority queue of AI research. Nonetheless, progress towards this achievement has been slow and incremental, with the problem remaining one of the many unsolved puzzles of the field. One crucial subproblem that must first be addressed is the development of an underlying *dialogue manager* capable of seamlessly coordinating several interacting elements: the interpretations of utterances that the system hears, the system’s own desires and beliefs, and other external events that happen in the world. Furthermore, the system must perform this coordination in a way that is *robust to unexpected events* and that is not inherently tied to any particular task - much like human dialogue participants are capable of.

This paper surveys various theoretical tools that have been proposed to model the way that humans and systems can interpret and produce dialogue - from *plans* to *schemas* - as well as a selection of notable dialogue manager implementations that in some way contribute to the goal of developing a fully general dialogue system. Furthermore, it introduces preliminary steps towards the creation of a novel dialogue manager that relies on dialogue schemas (symbolic representations of prototypical sequences of events in a dialogue), and deep understanding using a type-coherent, expressive logical form resembling surface English.

Contents

1	Introduction	3
2	Theories of Plan and Schema-Based Dialogue	5
2.1	Hierarchical Plans	6
2.1.1	STRIPS-like Operators	6
2.1.2	Revising Plans for Collaborative Dialogue	11
2.1.3	Incorporating Grounding and Discourse Obligations	17
2.2	Discourse Schemas	19
2.2.1	Schema Background	20
2.2.2	Discourse Schemas	21
2.2.3	Schema-Based Dialogue	22
3	Open-Domain Dialogue Systems	27
3.1	SharedPlan Architectures	28
3.1.1	COLLAGEN	28
3.1.2	Adapting for Health Counselling Dialogues	31
3.1.3	RavenClaw Dialogue Manager	33
3.2	TRIPS Dialogue System	33
3.2.1	TRIPS Architecture	34
3.2.2	Collaborative Problem Solving (CPS) Dialogue Shell	37
3.3	Information State Architectures	38
3.3.1	Conversational Practice Domains	39
3.3.2	Integrating Reward-Seeking Behavior	42
3.4	Incorporating Topic-Detection Techniques	44
3.4.1	Athena Dialogue Manager	44
3.4.2	Spoken Dialogue with Statistical Topic Detection	46
3.5	End-to-End Transformer-Based Dialogue Systems	47
3.5.1	Transformer Architectures for Dialogue	49
3.5.2	Effective Training and Decoding for Dialogue	49

3.5.3	Limitations	50
4	Towards a Schema-Based Open-domain System	52
4.1	Previous Work	53
4.2	Eta Dialogue Manager	54
4.2.1	Dialogue Schemas	54
4.2.2	System Architecture	57
4.2.3	Semantic Interpretation	59
4.2.4	Plan Execution	61
4.2.5	Response Generation	62
4.2.6	Task Queue	63
4.3	Virtual Human Applications using Eta	63
4.3.1	Blocks World Avatar (David)	64
4.3.2	Standardized Patient Avatar (SOPHIE)	66
4.4	Conclusion and Future Work	67

Chapter 1

Introduction

In some sense, creating a truly general, human-like spoken dialogue system is an “AI-hard” task: at least as hard as the hardest problems in AI. Such a system would require vast amounts of world knowledge across various domains; about both procedures and objects in the world. It would also require dynamically *acquiring* such knowledge. It would require advanced perceptual and interpretive capabilities, to transform auditory or imagistic inputs to latent symbolic (or sub-symbolic) representations. It would require extensive abilities to *reason* using those representations and stored knowledge, and the ability to *plan* or execute actions with the intent of achieving its own desires and objectives.

To reach a more achievable (though still lofty) goal, some of these problems can, and should, be abstracted away. What remains is the problem of *dialogue management*: how an automated system should *mediate* between its interpretations of the utterances of interlocutors, its own private cognitive state (goals, obligations, memories, knowledge, etc.), the context of the dialogue (facts that are assumed to be “common ground” between participants), and exogenous events that it observes in the world. Along these lines, Traum and Larsson, 2003 define a dialogue manager as consisting of the following (minimal) responsibilities:

1. Updating the dialogue context on the basis of the system’s interpretations of communication.
2. Using the dialogue context [as well as knowledge or memories possessed by the system] to guide interpretation of communication or observed external events.
3. Interfacing with domain-specific reasoners (e.g., a planner, ontology, or knowledge base) to coordinate dialogue and non-dialogue behavior and reasoning.
4. Deciding what action or utterance the system should perform next, and when it should perform it.

In this paper, I present an overview of *dialogue management* techniques for human-computer conversational systems, with a focus on *open-domain systems* that contribute in some way to the aforementioned goal of *general AI* - as opposed to systems that rely on limited frameworks applicable only in narrow tasks.

In Chapter 2, I discuss some of the theoretical prerequisites necessary to contextualize later dialogue manager implementations - these principally include the notions of **plans**, and **schemas**. Traditionally, theories of human dialogue have focused on the former concept - *plans*, i.e., sequences of actions that people take to achieve particular goals. Indeed, plans seem to take on first-order significance in human behavior: we can construct and execute them, simulate them in our minds, “debug” them, and infer plans of others (Cohen and Perrault, 1979). Additionally, we can communicate about plans and even “share” joint plans with other people.

However, in natural human conversation, people seem to be capable of *adaptive reasoning* beyond what this traditional view of plans can afford us. Turner, 1994 provides several characteristics of adaptive reasoning, including the ability to meld between both detailed planning and more “reactive” behavior, the ability to modify ongoing actions, to react to unanticipated events, and to seize opportunities. The human comportment of flexible, reactive behavior can be captured by the second concept: *schemas*, i.e., learned representations of “prototypical” patterns of events that people can construct and use to flexibly guide action, in a manner that is sensitive to unexpected events and that does not necessarily rely on detailed planning.

In Chapter 3, I discuss a collection of notable open-domain dialogue manager implementations that contribute in some way to the goal of developing a general dialogue manager which is domain-independent - in that one can have semantically broad conversations with it irrespective to any particular task - while also competent in applied tasks, i.e., those involving collaborative planning or domain-specific reasoning. Though no system has yet accomplished this, the last few decades have seen the development of a number of both *integrated systems* - end-to-end conversational systems competent in some applied task - and general *dialogue shells* - semantically broad domain-independent dialogue managers that are not “complete”, but are potentially usable as a sub-module in some practical system. I restrict my attention in this chapter to a handful of systems that have been impactful in the literature, and that appear to be a step towards a fully general dialogue manager.

Finally, in Chapter 4, I present my own preliminary efforts in this direction, predicated on the concept of *dialogue schemas* and the development of a dialogue manager that uses schemas as a basis for adaptive planning. I first describe the overall architecture of the dialogue manager being developed - Eta - and then discuss some virtual agents that have been developed using Eta, applied in specific practical domains. Some possible directions of future work are briefly highlighted.

Chapter 2

Theories of Plan and Schema-Based Dialogue

Before there were human-computer dialogue systems, there were attempts to explain and analyze human-human dialogue. Perhaps unsurprisingly, the former systems often had their roots in the models developed as a result of the latter efforts - the goal of a human-computer dialogue system is, after all, to replicate the richness of natural human conversation. Thus, before presenting a survey of human-computer dialogue systems, it is helpful to provide a conceptual base by first reviewing a selection of foundational theoretical works that contributed formal models of dialogue.

Most theoretical dialogue models to date have relied on *dialogue plans*: sequences of dialogue actions that are dynamically chained together by an agent in order to satisfy some dialogue goal, potentially subject to a changing dialogue context. Theoretical models of dialogue planning, in turn, can be characterized either as relying on reasoning over dialogue states (typically involving agents' internal cognitive states) to find a sequence of actions satisfying a goal state, or as relying on more reflexive or analogical reasoning based on learned patterns of behavior. In particular, planning systems reliant on the latter style of reasoning have often utilized the concept of *schemas*: structured representations capturing stereotypical patterns of events, objects, or concepts.

In this chapter, I first review the classical STRIPS state-space search planning formalism (Fikes and Nilsson, 1971) and discuss attempts to apply it to dialogue planning, drawing on theories of speech acts from linguistics. Next, I discuss a conceptual shift towards having plans represent internal cognitive attitudes of agents, and eventually towards collaborative plans that agents jointly enact. I then describe a generalization of this approach based on “information states”, which encompass various aspects of common ground as well as dialogue plans and cognitive states.

Finally, I introduce the notion of *schemas* in greater detail and discuss a precursory theory of *schema-based dialogue planning* developed by Roy Turner (Turner, 1994) that uses schemas in conjunction with context to form dialogue plans.

2.1 Hierarchical Plans

2.1.1 STRIPS-like Operators

The earliest theories of planning-based dialogue are in essence extensions of the STRIPS planning formalism (Fikes and Nilsson, 1971). A STRIPS planning problem defines a set of operators, or actions, which operate on states of the world. Theories of dialogue based on the STRIPS framework use operators to represent speech acts (for instance, for “request” or “inform” speech acts), and include in their representations of world states the individual cognitive states of dialogue participants, such as beliefs and intentions.

STRIPS Planning Formalism

Before showing how STRIPS-like operators can be used to model speech acts, I first briefly review the classical first-order STRIPS planning framework introduced by Fikes and Nilsson, 1971. Formally, a STRIPS problem is a tuple $\langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where:

- \mathcal{P} is a set of propositions, i.e. ground atomic formula from a first-order vocabulary.
- \mathcal{O} is a set of operators (or action types).
- $\mathcal{I} \subseteq \mathcal{P}$ is the initial state.
- $\mathcal{G} \subseteq \mathcal{P}$ is the goal state.

STRIPS employs a closed world assumption (CWA), such that for each state $s \subseteq \mathcal{P}$ and for any $P \in \mathcal{P}$, if $P \notin s$, then $\neg P$ is assumed to be true.

Furthermore, each operator $o \in \mathcal{O}$ is a tuple $o = \langle \text{HEADER}, \text{PRE}, \text{EFF} \rangle$, where:

- HEADER specifies an operator name and a subset of variables in the first-order vocabulary.
- PRE is a set of preconditions, i.e. literals whose arguments are constants in the first-order vocabulary or variables occurring in HEADER. Let PRE^+ and PRE^- be the set of atoms corresponding to positive and negative literals, respectively.

- EFF is a set of effects, i.e. literals whose arguments are constants in the first-order vocabulary or variables occurring in HEADER . Let EFF^+ and EFF^- be the set of atoms corresponding to positive and negative literals, respectively.

The effect of applying an operator o to a state s is defined by the successor function succ :

$$\text{succ}(o, s) = \begin{cases} (s \setminus \text{EFF}^-(o)) \cup \text{EFF}^+(o) & \text{PRE}^+(o) \subseteq s \text{ and } \text{PRE}^-(o) \cap s = \emptyset \\ s & \text{otherwise} \end{cases}$$

Applying a sequence of operators can be defined recursively as follows:

$$\begin{aligned} \text{succ}([], s) &= s \\ \text{succ}([o_1, o_2, \dots, o_n], s) &= \text{succ}([o_2, \dots, o_n], \text{succ}(o_1, s)) \end{aligned}$$

A STRIPS plan is therefore a sequence of operators $\Delta = [o_1, \dots, o_n]$ such that $\mathcal{G} \subseteq \text{succ}(\Delta, \mathcal{I})$. There are multiple algorithms for plan construction, but a typical algorithm is *backward chaining*: starting at \mathcal{G} , find an operator o that has \mathcal{G} as an effect, and then add unsatisfied preconditions of o as new goals, and repeat until the initial state is reached or some other termination criterion.

Two extensions to this formalism are commonly made, in order for it to be more readily applicable to real-life planning problems. First, plans often have a hierarchical structure: performing an “open door” action, for instance, might decompose into a bunch of sub-actions or sub-goals - moving towards the door, grasping the handle, etc. The operator then becomes a tuple $\langle \text{HEADER}, \text{PRE}, \text{EFF}, \text{BODY} \rangle$. BODY may be a sequence of operators $[o'_1, \dots, o'_n]$, or another goal state \mathcal{G}' to be achieved. As operator o_i in the top-level plan is executed, $\text{BODY}(o_i)$ is recursively executed (in the former case), or the plan construction algorithm is run on the new goals to find a subplan which is then executed (in the latter case).

The second extension concerns the fact that plans often aren’t strictly linear, but can consist of actions whose order may be flexible or interleaved. This is addressed by imposing a partial ordering on actions, rather than a total order. The modified plan construction algorithm induces a directed acyclic graph (DAG) where nodes consist of predicates and operators (and subplan graphs, in the hierarchical case), and edges represent the precondition and effect relations between nodes (and operator/body links, in the hierarchical case).

Adapting Operators for Speech Acts

A seminal work in dialogue theory by Cohen and Perrault, 1979 presents a relatively direct application of STRIPS-like operators to model dialogue as a sequence of speech acts.

The classical theory of speech acts developed by philosophers of language (Austin, 1962; Searle, 1969) posited that any *locutionary act* (an utterance of some sentence with definite meaning) corresponds to an associated *illocutionary act* (the act performed *in* making the utterance) and *perlocutionary act* (the act realized by performing the utterance). For instance, the locutionary act of uttering “It is cold in here.” to you might have an associated illocutionary act of *requesting you to close a window*, and an associated perlocutionary act of *convincing you to close a window*.

Searle, 1969 suggests several categories of necessary and sufficient conditions for the successful, or felicitous, performance of illocutionary acts. These conditions are presented below, illustrated using the example of speaker S uttering T (with propositional content p) to request that hearer H do action A.

- **Normal input/output conditions** include that H is not deaf, S is not joking, etc.
- **Propositional content conditions** state restrictions on the propositional content of utterances corresponding to specific speech acts. E.g., in expressing that p by utterance T, S predicates a future action of H.
- **Preparatory conditions** state preconditions for a speaker felicitously performing a speech act. E.g., H is able to do A; S believes that H is able to do A; it's not obvious to S and H that H will do A in the normal course of action.
- **Sincerity condition** states the condition for a speech act being sincere. E.g., S must want H to do A.
- **Essential condition** states the condition that a speech act must count as an act to achieve something. E.g., S intends that the utterance T will convince H to do A.
- **Reflexive intention condition** states that (i₁) S intends for H to believe that, by uttering T, S intends for them to do A; (i₂) S intends for H to recognize their intention i₁, and (i₃) S intends for H to be convinced to do A by means of recognition of i₁.

In order to translate Searle’s conditions to STRIPS-like operators, a first-order modal logic is assumed, with AGT BELIEVE P (henceforth abbreviated as BEL) and

```

INFORMREF(S, H,  $\lambda x D_x$ ):

CANDO.PR:  $\exists y S \text{ BEL } (ixDx)=y$ 
WANT.PR: S BEL S WANT informref-instance
EFFECT:  $\exists y H \text{ BEL } S \text{ BEL } (ixDx)=y$ 

```

Figure 2.1: Example of “informref” speech act operator from (Cohen and Perrault, 1979).

```

CONVINCEREF(AGT1, AGT2,  $\lambda x D_x$ ):

CANDO.PR:  $\exists y AGT \text{ BEL } AGT1 \text{ BEL } (ixDx)=y$ 
EFFECT:  $\exists y AGT \text{ BEL } (ixDx)=y$ 

```

Figure 2.2: Example of an operator allowing the perlocutionary effect of “informref”.

AGT WANT P denoting the propositions that AGT believes that P is true, and that AGT intends that P, respectively. A state in the planning model therefore includes the beliefs and intentions of each agent. The preconditions of operators are divided into WANT.PR preconditions and CANDO.PR preconditions, which represent the “non-obviousness‘ preparatory condition and any remaining preparatory conditions of the speech act, respectively. The EFFECT of the operator reflects the essential condition of the speech act (i.e., S believes that S intends some instantiation of the speech act). Other conditions of Searle’s, Cohen argues, emerge out of the process of planning itself and are not represented in the particular speech act operators.

Furthermore, to enable complete planning, the perlocutionary acts must be modelled as operators as well - for instance, the `inform` speech act produces in the hearer a belief that the speaker believes some proposition, but an accompanying `convince` operator must be added to reach the state where the hearer believes the proposition.

Examples of operators for `informref` (informing a hearer of the value of some definite description, such as in a ‘wh-question’) and `convinceref` are shown in Figures 2.1 and 2.2, respectively. The preconditions and effects of these operators contain free variables which get ‘filled in’ as the plan is constructed. Note that “`ixDx`” represents the definite description “the x which is D”, where D is a predicate with one free variable.

Further details of this theory are developed by Allen and Perrault, 1980, who provide a tractable algorithm for collaboratively constructing and executing dialogue plans as a conversation proceeds. The logical representation and operator syntax are

the same as in (Cohen and Perrault, 1979), except that operators are augmented with **body** decompositions, thereby resulting in the DAG plan structures described in the overview of STRIPS and its extensions. Operators may also specify a set of constraints on the instantiation of the parameters occurring in the header of the operator.

For a goal-oriented dialogue to proceed, an agent will often have to infer the plans, beliefs, and goals of the other agent before continuing their own plan. This inference process is modelled as a search through a set of partial plans, each represented as tuples $\pi = \langle \Delta_{\text{alt}}, \Delta_{\text{exp}}, r \rangle$, where:

- Δ_{alt} is an *alternative* plan graph, created by chaining *plan inference rules* from an observed action by the other agent.
- Δ_{exp} is an *expected* plan graph, created by chaining *plan construction rules* from an expected goal of the other agent.
- $r \in \mathbb{R}$ is a rating for the partial plan π , initialized to 1 and updated using a set of heuristics.

A non-exhaustive list of examples of plan inference rules are provided in eqs. (2.1) to (2.3), and plan construction rules in eqs. (2.4) to (2.6). Application of each rule yields the addition of an additional node and arc in the corresponding plan graph.

[Precondition-Action Rule]	$S \text{ BEL } A \text{ WANT } P \Rightarrow S \text{ BEL } A \text{ WANT } ACT$	
	if P is a precondition of ACT	(2.1)
[Body-Action Rule]	$S \text{ BEL } A \text{ WANT } B \Rightarrow S \text{ BEL } A \text{ WANT } ACT$	
	if B is part of the body of ACT	(2.2)
[Action-Effect Rule]	$S \text{ BEL } A \text{ WANT } ACT \Rightarrow S \text{ BEL } A \text{ WANT } E$	
	if E is an effect of ACT	(2.3)
[Action-Precondition Rule]	$X \text{ WANT } ACT \Rightarrow X \text{ WANT } P$	
	if P is a precondition of ACT	(2.4)
[Action-Body Rule]	$X \text{ WANT } ACT \Rightarrow X \text{ WANT } B$	
	if B is part of the body of ACT	(2.5)
[Effect-Action Rule]	$X \text{ WANT } E \Rightarrow X \text{ WANT } ACT$	
	if E is an effect of ACT	(2.6)

Note that the plan construction rules are simply reciprocals of the plan inference rules (with the caveat that the belief operators may be dropped, as plan construction rules represent the process by which an agent constructs their own plan, rather than beliefs about the plans of others). Indeed, the underlying intuition behind the plan inference algorithm is that this process can be made tractable by simultaneously simulating both forward chaining from an observed action and backward chaining from an expected goal, finding pairs that ‘meet in the middle’ with unifiable actions or subgoals, and prioritizing the expansion of partial plans which are more likely to efficiently lead to a unification.

Examples of heuristics used to assign ratings in the model include rules such as:

- Decrease the rating of a partial plan if it contains an action whose preconditions are false at the time of the action’s execution.
- Increase the rating of a partial plan if its alternative contains propositions which are unifiable with propositions in its expectation.

At each step of the plan inference algorithm, the partial plan with the highest rating is selected and an appropriate “task” is selected - the primary tasks are identifying applicable inference rules for the partial plan, expanding the partial plan according to the identified inference rules, finding referents for descriptions in plans, or accepting a partial plan and terminating plan inference. The plan inference process yields a plan graph which can then be executed by the agent.

A full treatment of the various inference rules, heuristics, and control of the plan inference process used in Allen’s model will not be presented here; interested readers may find those details in (Allen and Perrault, 1980).

2.1.2 Revising Plans for Collaborative Dialogue

Though the classical STRIPS-based theories of dialogue presented a powerful tool for analysis of task-specific goal-oriented dialogues, they faced major limitations for analyzing many types of collaborative dialogues:

- These models assumed that the inferring agent has both complete and correct knowledge of available actions - this prevents analyses of dialogues where one agent may have an invalid plan, and the other agent has to issue a correction to them.
- A certain asymmetry in dialogue was also assumed: the speaker has private desires and produces utterances, while the hearer attempts to infer the speaker’s beliefs and intentions from the utterance and use them to construct their own

plan. Grosz and Sidner, 1990 term this assumption the “master-slave assumption”, and make the point that collaboration typically isn’t merely the combination of the individual plans of two or more agents, but rather is the result of construction and execution of a joint plan.

Plans as Cognitive Attitudes

The first of these limitations was addressed by Pollack, 1990, who considered dialogues where participants attempted to communicate about potentially invalid plans. For instance, a speaker might request the phone number of a hospital, under the impression that her sibling is there, only to be corrected by the hearer that the sibling has been discharged and can be reached at their home number. The assumption that available plan operators are shared and mutually understood to both participants fails in this case, and so Pollack proposes a recalibration: treating plan operators as themselves private cognitive attitudes, with beliefs about plan operators that can be correct or incorrect.

Pollack first notes that for agent A to have a plan to do act β , that consists of a set of acts Δ , is really to have the following set of conditions:

1. A believes that A can execute each act $\alpha \in \Delta$.
2. A believes that executing the acts $\alpha \in \Delta$, in their (possibly partial) temporal order, will entail A’s performance of β .
3. For each $\alpha \in \Delta$, A believes that by doing α , he will contribute to doing β or some other act γ that plays a role in the plan.
4. A must intend to execute each act $\alpha \in \Delta$, in their (possibly partial) temporal order.
5. A must intend to execute Δ as a way of doing β .
6. For each $\alpha \in \Delta$, A must intend by doing α to contribute to doing β or some other act γ that plays a role in the plan.

To further operationalize the conditions in 2 and 6, a more careful definition of “contributing to an action” is required. Pollack considers two senses in which α might contribute to β . First, α might *generate* β , in that *by* doing α , A would have done β . For example, by uttering “It is cold in here” under certain conditions, A would have generated the act of requesting a window to be closed. Second, α might *enable* β . Though A searching a phone book for B’s number might enable A calling B, it is certainly not the case that the former generates the latter.

Pollack focuses primarily on *SimplePlans*, which are assumed to only have *generate* relations between acts within the plan, and no *enable* relations. Furthermore, a first-order temporal logic (Allen, 1984) is assumed, with predicates $\text{HOLDS}(P,t)$ and $\text{OCCURS}(\alpha, A, t)$ denoting respectively that P holds throughout time t , and that A performs act α during time t . The following predicates are additionally defined (the former two have logical definitions given in the paper, while the remainder are primitives):

- $\text{CGEN}(\alpha, \beta, C)$: α conditionally generates β under certain conditions C .
- $\text{GEN}(\alpha, \beta, A, t)$: A 's doing α at time t generates A 's doing β .
- $\text{EXEC}(\alpha, A, t)$: A 's doing α at time t is executable.
- $\text{BEL}(A, P, t)$: A believes P throughout time t .
- $\text{INT}(A, \alpha, t_1, t_2)$: Throughout time t_1 , A intends to do α at time t_2 .

Finally, a function $\text{by}(\alpha, \beta)$ is also defined, which maps its arguments to the act of doing β by α . With this in hand, Pollack refines the previous conditions to define A having a SimplePlan at time t_1 to do α_n , consisting of doing acts $[\alpha_1, \dots, \alpha_{n-1}]$ at time t_2 :

$$\begin{aligned} \text{SIMPLE-PLAN}(A, \alpha_n, [\alpha_1, \dots, \alpha_{n-1}], t_2, t_1) \iff & \\ (1) \text{ BEL}(A, \text{EXEC}(\alpha_i, A, t_2), t_1) \text{ for } i = 1, \dots, n \wedge & \\ (2) \text{ BEL}(A, \text{GEN}(\alpha_i, \alpha_{i+1}, A, t_2), t_1) \text{ for } i = 1, \dots, n - 1 \wedge & \\ (3) \text{ INT}(A, \alpha_i, t_2, t_1) \text{ for } i = 1, \dots, n \wedge & \\ (4) \text{ INT}(A, \text{by}(\alpha_i, \alpha_{i+1}), t_2, t_1) \text{ for } i = 1, \dots, n - 1 \wedge & \end{aligned} \tag{2.7}$$

The primary insight behind this new plan definition is that it allows us to talk precisely about what it means for an agent A to have an invalid plan (as in the phone number example above): A has the set of beliefs and intentions given by the definition of **SIMPLE-PLAN**, and one or more of the beliefs is incorrect, or the corresponding intentions unrealizable. Furthermore, A can believe that another agent B has an invalid **SIMPLE-PLAN** if A attributes to B beliefs about the relations between actions in a plan which A believes to be false. Unlike the STRIPS-based models, these facts allow us to analyze conversations that deviate from “perfect” dialogue behavior where agents operate according to mutually understood, well-formed plans. Modelling plans as complex cognitive attitudes allows us, for instance, to analyze dialogues where an agent issues a correction to another, or suggests some sort of plan modification.

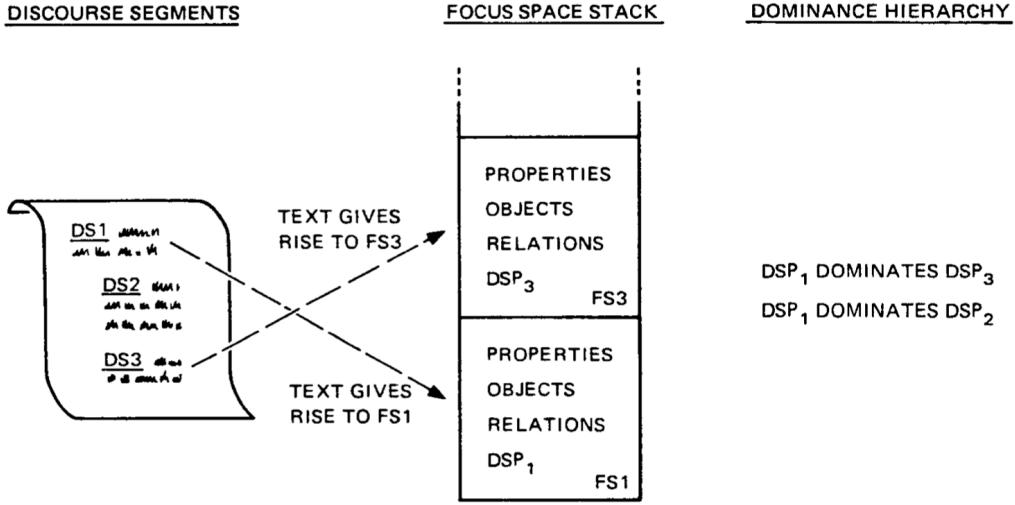


Figure 2.3: The three components in the tripartite model of discourse structure.

SharedPlan Approach

Though Pollack's work addressed the first of the aforementioned limitations, her plan model ultimately still required hearers to reason about and adopt the goals and beliefs of speakers in a way which is unrealistic in many collaborative dialogues. The SharedPlan formalism attempts to address some of these limitations. Though the SharedPlan theory is similar to the previous theories in that it employs a logical planning approach and focuses primarily on reasoning about agents' mental states, it "differs significantly from the process of reasoning with plan operators" (Lochbaum, 1998): rather than each agent merely building and reasoning about individual plans, the SharedPlan approach views dialogue participants as collaboratively building and augmenting a joint plan by contributing information about their beliefs, wants, and individual plans through utterances.

The SharedPlan theory was developed and refined over the course of many publications (Grosz and Sidner, 1990; Grosz and Kraus, 1996; Lochbaum, 1998); in the following presentation, I will focus on the theory as outlined by Lochbaum as it presents the most complete version of the theory.

At the core of the SharedPlan approach is a tripartite model of discourse structure first introduced in Grosz and Sidner, 1986, consisting of a hierarchical structure of utterances in the discourse, a "dominance hierarchy" of discourse-level intents (i.e., discourse purposes or DSPs) corresponding to discourse segments, and an attentional state (represented as a stack of focus spaces, with each focus space containing relevant

properties, objects, relations, as well as the discourse-level intent for that discourse segment). These three components are illustrated in Figure 2.3. The components in this model, particularly the dominance hierarchy of discourse-level intents, directly correspond to relations among SharedPlans for dialogue acts.

Agents involved in a conversation can have Individual Plans (IP), defined analogously to the SIMPLE-PLAN definition provided above, as well as Shared Plans (SP). Each of these types of plans receives a full definition in Allen’s temporal logic (Allen, 1984); for brevity’s sake, only the high-level description of a SharedPlan will be given.

The conditions that must be met for a group of agents GR to have a SHARED-PLAN for α are as follows:

0. GR is committed to performing α
1. GR has a recipe for α
2. For each single-agent constituent act β_i in the recipe, there is an agent $G_{\beta_i} \in GR$ such that:
 - (a) G_{β_i} intends to perform β_i
 G_{β_i} believes that it can perform β_i
 G_{β_i} has an individual plan for β_i
 - (b) The group GR mutually believe (2a)
 - (c) The group GR is committed to G_{β_i} ’s success
3. For each multi-agent constituent act β_i in the recipe, there is a subgroup of agents $GR_{\beta_i} \subseteq GR$ such that:
 - (a) GR_{β_i} mutually believe that they can perform β_i
 GR_{β_i} has a SharedPlan for β_i
 - (b) The group GR mutually believe (3a)
 - (c) The group GR is committed to GR_{β_i} ’s success

In the above definition, a “recipe” for α is a set of constituent acts $\{\beta_1, \dots, \beta_n\}$, parameters, and constraints. The truth of the ability-related conditions (e.g., (2a) and (3a)) depends crucially on whether the agent has a recipe for an action, whether it’s able to identify the parameters for the action, and whether it’s able to satisfy the constraints of the recipe.

Agents can also have *partial* Individual Plans or Shared Plans (PIPs or PSPs, respectively), which are IPs or SPs in which only a subset of the necessary conditions

hold. Plans in which all conditions hold are also called *full* Individual Plans (FIPs) or SharedPlans (FSPs). The process of collaborative reasoning with SharedPlans involves *augmenting* a partial plan to build a full plan, with individual utterances contributing to the augmentation process. Assuming that agents G_1 and G_2 already have a PSP to do α , if G_1 observes G_2 make utterance U communicating proposition P, then G_1 will assume $\text{MB}(\{G_1, G_2\}, \text{BEL}(G_2, P))$, where MB is a predicate denoting mutual belief. Subsequently, G_1 will attempt to determine the relation of P to the current SharedPlan based on the following conditions:

1. If G_1 believes that U or P indicates the initiation of a subsidiary SharedPlan for an act β , then G_1 will assume $\text{Int.Th}(G_2, \text{FSP}(\{G_1, G_2\}, \beta))$ (where Int.Th denotes an intention by an agent that something be true, and FSP denotes that an agent has a full SharedPlan for act β), and decide whether he is also willing to adopt the same intention.
2. If G_1 believes that U or P indicates the completion of the current SharedPlan, then G_1 will assume $\text{BEL}(G_2, \text{FSP}(\{G_1, G_2\}, \alpha))$, and determine if he has the same belief.
3. Otherwise, G_1 will ascribe to G_2 a belief that P is relevant to the current SharedPlan, and determine if he also believes that to be the case.

If the above is successful, then G_1 will signal his agreement (possibly implicitly) and augment his PSP.

We can now indicate the link between SharedPlans and the DSPs in the discourse model more precisely: if DSP_1 dominates DSP_2 , then there's a subsidiary relationship between the SharedPlans corresponding to these discourse segments. Similarly, completed discourse segments have full SharedPlans associated, while incomplete ones have partial SharedPlans associated. Plan augmentation, as well as manipulations of the focus stack, are therefore “reduced” to recognizing whether the user’s utterance or proposition initiates, completes, or contributes to a discourse segment. Lochbaum describes how this process may be implemented through a set of *conversational default rules* (i.e., inference rules that apply in absence of evidence to the contrary) and a *recipe graph* constructed by composing individual recipes (essentially, production rules with associated constraints).

In closing, Lochbaum notes several directions of future research: first, the plan augmentation process needs to be further researched and developed; the exposition given in the paper, though formal, was incomplete. Second, the analysis assumed that DSPs always involved SharedPlans, though this need not always be the case. Finally, there is the question of the utility of the SharedPlans theory for creating

practical collaborative systems - a topic that is discussed further in Section 3.1, where I discuss dialogue systems created using the SharedPlans formalism.

2.1.3 Incorporating Grounding and Discourse Obligations

Though the SharedPlans approach addresses some limitations of the pure plan-based approach, it still requires a strong degree of collaboration and reasoning in dialogue which restricts its applicability. Other factors may be more apt in explaining many ‘day-to-day’ interactions. Traum and Allen, 1994, for instance, discuss the role of social discourse obligations: an agent asked “Do you have the time?” by a stranger likely doesn’t establish a shared plan or reason extensively about the stranger’s beliefs and goals, but rather simply reacts according to a learned social obligation or convention related to such requests.

Perhaps more gravely, many aspects of ordinary conversation (including the collaborative kind) are difficult or impossible to analyze using classical theories of dialogue planning. Utterances do not merely represent speech acts, but often comprise of multiple actions at the sub-utterance level related to turn-taking, repairs, backchannels (“yeah”, “I see”, etc.), etc. Furthermore, a classical speech act such as a request may not necessarily correspond to a single utterance, but may consist of a larger *discourse unit*, such as an “adjacency pair” consisting of a proposal and acknowledgement. Such discourse units are often necessary for the successful *grounding* information, i.e., adding a piece of information, and the mutual understanding thereof, to the common ground (Clark and Brennan, 1991; Clark, 1996). In fact, the role of common ground in the previous theories has typically been left implicit and minimal - consisting of the observation of speech acts by both participants, and the reflexive inferences about beliefs and intentions which follow from that.

The *Information State* approach to dialogue (Poesio and Traum, 1998; Traum and Larsson, 2003) provides a way to explicate the concept of common ground, and to account for non-plan-based utterances (e.g., those based on discourse obligations). An information state consists of “wrapping” the cognitive states and plans of the previous theories, along with other aspects of the dialogue such as common ground/context, linguistic structure, obligations and commitments, etc., into a single dialogue representation, which gets updated according to particular rules. The initial formulation of the information state approach by Poesio and Traum, 1998 relied on Discourse Representation Structures (DRSs) (Kamp et al., 2011), but this was later generalized to provide a more flexible framework. An Information State dialogue framework consists of the following components:

1. An **information state** consisting of informational components (participants, common ground/context, discourse structure, previous move(s), obligations

and commitments, beliefs, intentions, user models and individual plans, etc.). These are represented formally using lists, sets, typed structures, records, DRSs, propositions or modal operators, etc.

2. A set of **dialogue moves** that update the information state, and rules for recognizing/realizing performance of these moves (for instance, those provided by contemporary natural language understanding and generation systems).
3. A set of **update rules** governing how the information state is updated, given the conditions of the current information state and any observed dialogue moves. Update rules have a set of preconditions specifying conditions on the values of particular components of the information state (possibly including variables to be unified), and a set of effects to be applied to particular components of the information state (possibly using any variables bound by the preconditions).
4. An **update strategy** for deciding which rule(s) to apply from a set of applicable ones. Can range from straightforward strategies such as choosing the first applicable rule, to more complex strategies such as choosing rules according to assigned probabilities.

Note that the information state may contain both shared/public components (e.g., the common ground, participants, discourse structure and history, etc.) and private components (e.g., individual beliefs and intentions), allowing it to subsume some of the previous mental state-based theories that were discussed. The model can also contain static domain knowledge, which may be used in checking the preconditions of update rules.

A simple toy example of this approach is shown in Figure 2.4, with a set of update rules being shown on the left, and the analysis of a question-answer pair being shown on the right. This example consists of a very simple information state containing sets of private and shared beliefs, private agendas (stacks of intentions), and shared knowledge of the previous move and a stack of “questions under discussion” (QUD) - all initially empty except for the private agendas.

In step (1), the only applicable rule is `selectAsk`, which selects an *ask* speech act as the expected next move given the agenda of the turn-holding agent (in this case, a computer system). As the agent makes the question utterance, the `integrateSysAsk` rule in step (2) pops the agent’s agenda stack, and pushes the question that was asked onto the QUD stack. Under this context, any utterance the user makes is assumed to be an *answer* speech act, and if the utterance passes certain domain checks (such as relevance to the question), the propositional content of the answer is integrated into common ground in step (3) via `integrateUserAnswer`. Finally, since

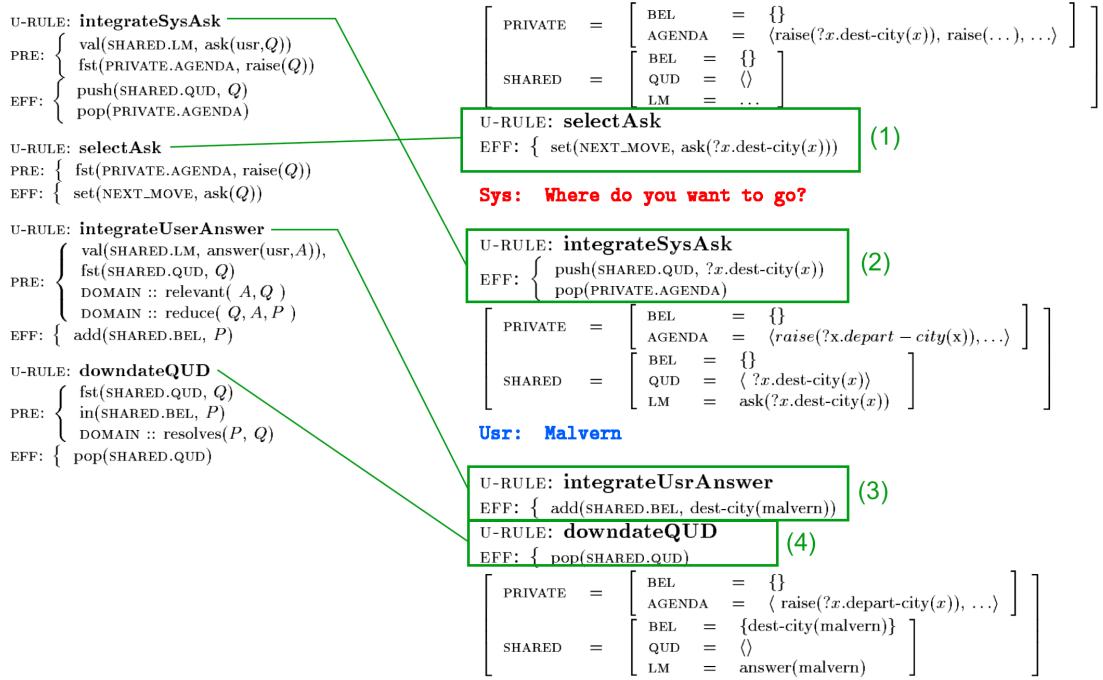


Figure 2.4: An information state is updated according to a set of update rules as a dialogue proceeds (Traum and Larsson, 2003). System utterances are shown in red, and user utterances in blue. Application of update rules are shown in green.

the propositional content in common ground resolves the question under discussion, the QUD stack is popped in step (4) using `downdateQUD`.

Although this example is simplistic, the representation is quite flexible and can be extended with additional constructs such as those discussed previously. It provides a powerful and transparent framework for dialogue analysis or management, and has been used to create task-specific dialogue systems (Larsson et al., 2002; Matheson et al., 2002) as well as later systems aimed at more open-ended domains, such as those discussed in Section 3.3.

2.2 Discourse Schemas

Though plan-based models, including those embedded within an “information state”, can be used to analyze a wide range of collaborative and task-oriented dialogues, these models still face challenges in accounting for the robustness of everyday human dialogue where discursive behavior and unanticipated events are common. In a system

where the information state is modified according to update rules, designing a set of rules sufficiently general to handle such unanticipated events remains a difficult problem, requiring significant hand-engineering efforts (see discussion in (Morbini et al., 2014), for instance). More generally, making possible changes in contextual knowledge endogenous to the dialogue planning process (i.e., as conditions/effects within rules or operators) imposes a trade-off between an agent exhibiting the same behavior regardless of context on one extreme, and on the other extreme resulting in a combinatorial explosion of rules/operators with each instance conditioned on a different context (Wilson and Turner, 2017).

An alternative approach to modelling dialogue relies on the concept of “schemas”. Schemas are similar to hierarchical plans, but representing descriptions of *stereotypical* events or situations, rather than recipes that directly entail sequences of actions necessary to accomplish some goal (in fact, hierarchical plans can be seen as a special case of schemas (Turner, 1994) where a goal has been explicitly specified, and every step within the schema specifies an act by a participant). The prototypical nature of schemas makes them more amenable to unexpected events and “on-the-fly” modifications to the dialogue. It also allows a schema-based model to find a flexible middle ground between rigid but thorough plan-based behavior, and more “reactive” but robust behavior. In this section, I provide some background on schemas, and review some previous work which uses schemas in dialogue processing. In Chapter 4, I discuss my group’s specific schema language.

2.2.1 Schema Background

The concept of a “schema” has lived a relatively interdisciplinary history, having first been popularized in psychology as a construct for analyzing cultural variation in stereotypical perceptions (Bartlett and Kintsch, 1995), and later adapted by cognitive science and artificial intelligence researchers as a proposed means of representing general prototypical knowledge about events and objects. In particular, Schank and Abelson, 1977 introduced *scripts* to represent prototypical patterns of events (such as the sequence of steps and conditions typically involved in going to a restaurant), while Minsky, 1974 introduced *frames* to represent prototypical objects or concepts. The concept of *schemas* is understood to encompass both types of prototypical representations, which are sometimes also referred to as *event schemas* and *object schemas* respectively (the latter terminology will be adopted in this paper).

While the specific representation of schemas varies across theories, they are generally characterized by several properties they share in common: first, the information they represent is prototypical and abstract. An actual instance of an event or object need not satisfy all properties in a schema to be “matched” to that schema. Part

of the power of this representation is that a partial schema match of an observation can be used to abductively infer other properties of that observation - for instance, observing a person in a bank with a gun might trigger a “bank robbery” schema, causing the observer to infer other properties associated with the event, such as that the robber intends to steal money. These inferences are fallible, however, and partial schema matches may be discarded as other schemas become more appropriate - for instance, the person with a gun may turn out to be a security guard.

Second, schemas may be hierarchically organized at different levels of granularity. For instance, a “going to restaraunt” schema may have associated subschemas, such as “going to a buffet”, “getting fast food”, etc. These lower-level schemas inherit the properties of the subsuming schema, though may also override particular properties (for instance, a fast food restaurant will typically not have waiters).

The latter property also implies that the processes of *generalization* and *specialization* likely play an important role in how humans learn and modify schemas to begin with - with people forming new schemas as they encounter similar experiences that warrant generalization. Though the topic of schema learning lies outside the scope of this paper¹, it’s important to note the appeal of having a model of dialogue whose components are *learnable* by analyzing corpora of dialogue, while still allowing for goal-directed behavior.

2.2.2 Discourse Schemas

One of the earliest attempts to use schemas in discourse analysis appeared in (Van Dijk and Kintsch, 1983). The model presented assumes that participants in a discourse actively construct mental representations (in some intensional logical form) of the events being communicated by a speaker, and in turn use these representations to generate utterances. The construction of these mental representations is governed by a set of “strategies” that concern multiple levels of understanding - from interpreting utterances as propositions, to establishing local coherence between utterances, to forming a *macrostructure* of the discourse.

A macrostructure is a hierarchical semantic description of a discourse, with the lowest level consisting of the utterance-level (or sub-utterance-level) propositional forms, and the highest level representing the topic or global meaning of the discourse. The model describes higher levels of the macrostructure as being derived from sequences of propositions in lower levels through three context-dependent macrorules:

1. **DELETION:** Delete each proposition that is not an interpretation condition (e.g., a presupposition) for another proposition in the sequence.

¹See (Lawley et al., 2019) for more details.

2. **GENERALIZATION:** Substitute the sequence by a proposition that is entailed by each proposition in the sequence.
3. **CONSTRUCTION:** Replace the sequence by a proposition that is entailed by the joint set of propositions in the sequence.

If macrostructures describe the global semantic structure of a *particular* discourse, *schematic structures* (or “superstructures”) are analogous to conceptual *patterns* of macrostructures - for instance, the macrostructures of rhetorical arguments are often similar to each other in form even if they differ in content; likewise for story narratives, etc. The form of a schematic structure mirrors those of macrostructures, but contains categories that act as *global functions* to specific macropropositions in a particular macrostructure. In this way, a subset of the categories of a schematic structure can be “matched” to macropropositions as a discourse proceeds, allowing other categories to be inferred, and imposing further constraints on interpretation. The relation between schematic structures and macrostructures also acts in the opposite direction: the construction of new macrostructures can serve to generalize, specialize, or modify existing schematic structures.

Van Dijk and Kintsch sketch how various types of contextual and schematic knowledge combine to produce speech acts and dialogue; this diagram is shown in Figure 2.5. Though this work was widely influential and contributed powerful insights into the role of schematic knowledge in discourse understanding, the ideas presented were fairly high-level and difficult to formalize in a computational system.

The concept of schematic knowledge is also applied to dialogue analysis in (Poesio and Traum, 1997), albeit in the narrower form of *discourse scripts* - in the same vein as Schank’s procedural event scripts. In this model, discourses follow a “thread” - an event that decomposes into constituent events, represented using tools from Discourse Representation Theory (DRT) (Kamp et al., 2011). A *discourse script* represents stereotypical knowledge that people have about certain kinds of threads. Similarly to Schank’s scripts, if one can recognize the discourse script that a given speech act is a part of - for instance, by matching previous speech acts, or contextual features - one can predict future speech acts, and use these expectations to constrain the interpretation of subsequent utterances.

2.2.3 Schema-Based Dialogue

An early attempt to address the challenge of handling unexpected events using a schema-based approach to dialogue was due to Turner, 1994. The proposed model relies on the notion of *context-mediated behavior*, wherein possible contexts are themselves first-class objects represented as schematic knowledge.

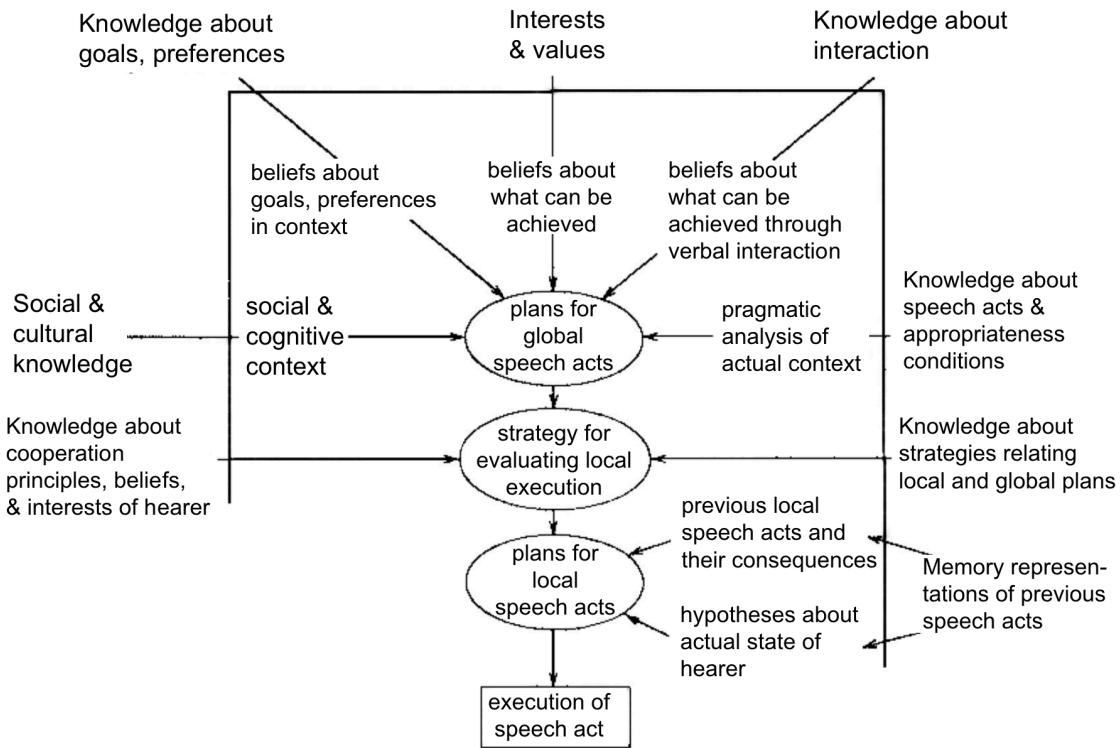


Figure 2.5: A diagram of the theory of dialogue presented by Van Dijk and Kintsch, 1983.

More precisely, the author introduces three subtypes of schemas:

- **Procedural schemas (p-schemas)** consist of an actor, a goal, and a set of steps. Each step may be a primitive action, a subgoal, or another p-schema. A p-schema may be specialized or generalized, are interruptable/resumable, and are only expanded as much as necessary.
- **Contextual schemas (c-schemas)** represent specific contexts or situations an agent may encounter, and encode domain-dependent aspects of that context (such as information about responding to events). Specifically, a c-schema can have the following components:
 - *Features* of the context used to determine when the c-schema is applicable.
 - *Standing orders* that cause the agent to perform certain acts automatically when a c-schema is selected.

- *Event information* mapping specific event occurrences to new goals that should be adapted in response.
- *Event importance thresholds* for specific events determining how reactive the system should be to those events within a particular context.
- *Attention focusing information* that determines which goals an agent should attend to when in a particular context.
- *Action selecting information* linking goals to specific actions (p-schemas) that can be used to achieve those goals when in a particular context.
- **Strategic schemas (s-schemas)** represent information about an agent's domain-independent problem-solving strategies across contexts. An s-schema can set the agent's *goal importance* information, i.e. order different types of goals by precedence, as well as set an overall *event importance threshold* determining how sensitive the agent is to reacting to unexpected events in general.

Examples of these three types of schemas are shown in Figure 2.6. Though the c-schema is taken from a non-dialogue domain (an autonomous underwater vehicle (AUV) domain where the context is the AUV being within a harbor), the representation that it illustrates is consistent with that used in dialogue-based domains.

The schema-based reasoning process allows for dynamic modification of a dialogue or plan as exogenous events are observed. This process is diagrammed in Figure 2.7. Only one s-schema is assumed to be active at a time, but multiple c-schemas may be active simultaneously. Starting from step (1), the reasoner is initialized with a list of active goals. In step (2), the reasoner focuses attention on a particular goal using the s-schema's *goal importance* information, and the c-schema's *attention-focusing* information. In step (3), a p-schema is selected conditionally on the chosen goal and the *action selecting information* in the c-schema mapping goals to actions within that context. In step (4), the p-schema is applied while simultaneously updating the environment and monitoring for external changes. If no external change is detected, the process returns to step (2) and repeats; otherwise the reasoner proceeds through the event handler component. The change is first diagnosed/classified in (5a), and assessed as important or unimportant using the c-schema and s-schema's *event importance thresholds* in (5b). If a change is deemed important, it is used to select a new goal according to the c-schema's *event information*, which is added to the list of active goals. The process then repeats at step (2).

A separate process is also used to select an appropriate s-schema and c-schemas based on observation of the current situation. The author does not give a detailed description of the process of retrieving schemas from memory, but notes that any schema memory capable of dynamically adding/removing schemas, or changing the

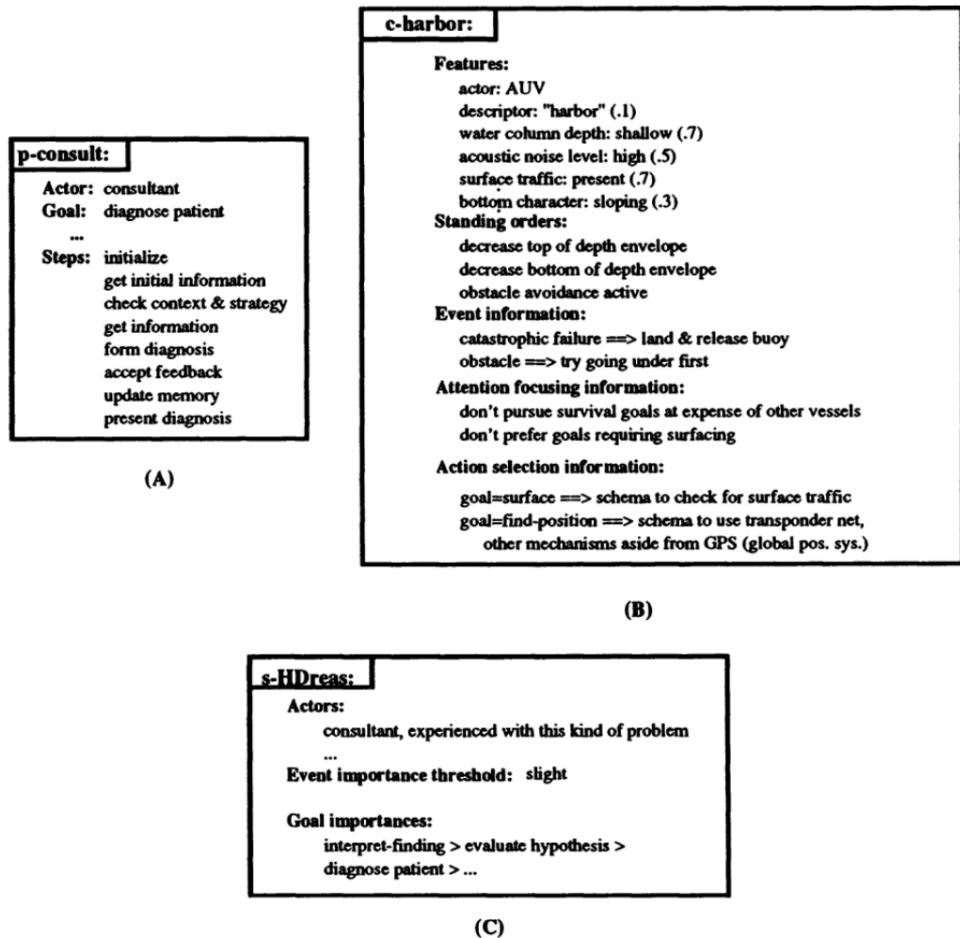


Figure 2.6: Examples of a p-schema, c-schema, and s-schema from Turner, 1994. The p-schema and s-schema are taken from a medical consulting domain, while the c-schema is taken from an AUV planning domain.

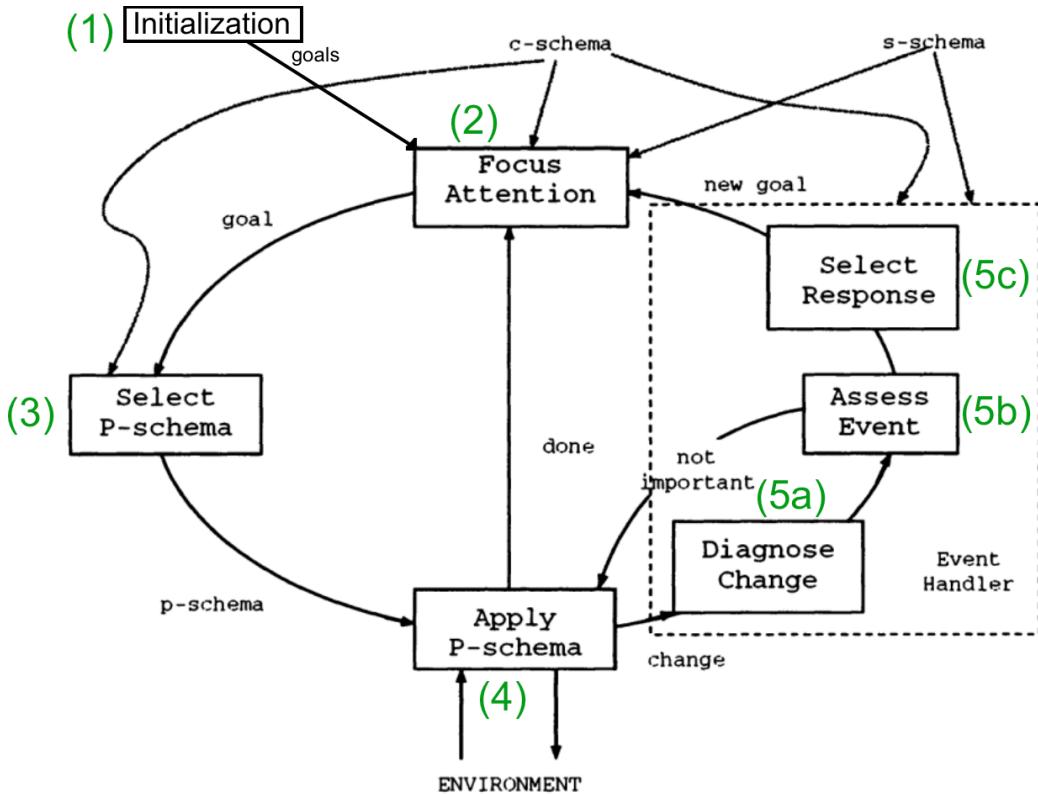


Figure 2.7: The schema-based reasoning process diagrammed by Turner, 1994, with annotated steps.

way that they're organized, can be used. The model presented above uses Schank's dynamic conceptual memory (Schank and Burstein, 1985).

This schema-based dialogue model demonstrates several features which will ultimately be important in future schema-based systems - for example, an algorithm to dynamically fetch candidate schemas whenever external observations match certain conditions within those schemas. However, the schema language used by the model is greatly limited, relying on simple propositional slot values rather than semantically rich logical representations. Furthermore, it's questionable to what extent certain contextually-relevant actions by the user should be *separated* from the concept of a procedural schema (or more generally, an episode schema). Ordinary procedures, such as the classic Schankian "restaraunt" example, often do encompass conditional behavior associated with the procedure (e.g., whether or not to leave a tip), rather than any inherent aspect of the context.

Chapter 3

Open-Domain Dialogue Systems

In this chapter, I turn my attention towards reviewing seminal efforts in creating *open-domain*, mixed-initiative dialogue systems. These systems are distinguished from dialogue systems reliant on task-specific finite-state transition systems (McTear, 1998) or slot-filling approaches (Xu and Rudnicky, 2000), which tend to be useful in narrow, terse, system-initiated dialogues, but too rigid for open-domain dialogues. On the other hand, the open-domain systems discussed here are also distinguished from chatbots (such as Weizenbaum's ELIZA (Weizenbaum, 1966) and contemporary deep learning-based analogues) which, while capable of topically broad discussion, are not capable of goal-directed behavior and thus are generally insufficient for many practical open-domain applications. Nevertheless, I do discuss a state-of-the-art deep learning-based dialogue system in Section 3.5 that uses specialized models and training procedures to achieve some degree of local coherence, making it potentially useful in some narrow open-domain applications (e.g., conversational practice systems) or as a sub-component in a dialogue manager.

With the exception of the end-to-end deep learning system, most of the dialogue systems presented in this chapter are rooted in one of the theoretical models discussed in Chapter 2. This is perhaps unsurprising, as the ultimate goal of an open-domain human-computer dialogue system is to achieve the level of linguistic complexity and dynamicity present in the natural human-human dialogues that those models were developed to explain. Though many of the systems presented in this chapter are applied or tested in specific domains, they are still distinguished by an underlying dialogue model which is capable of being extended to more general domains (in the ideal case, just requiring an augmentation of declarative world knowledge - the automation of which is an as-of-yet unsolved task in AI).

3.1 SharedPlan Architectures

3.1.1 COLLAGEN

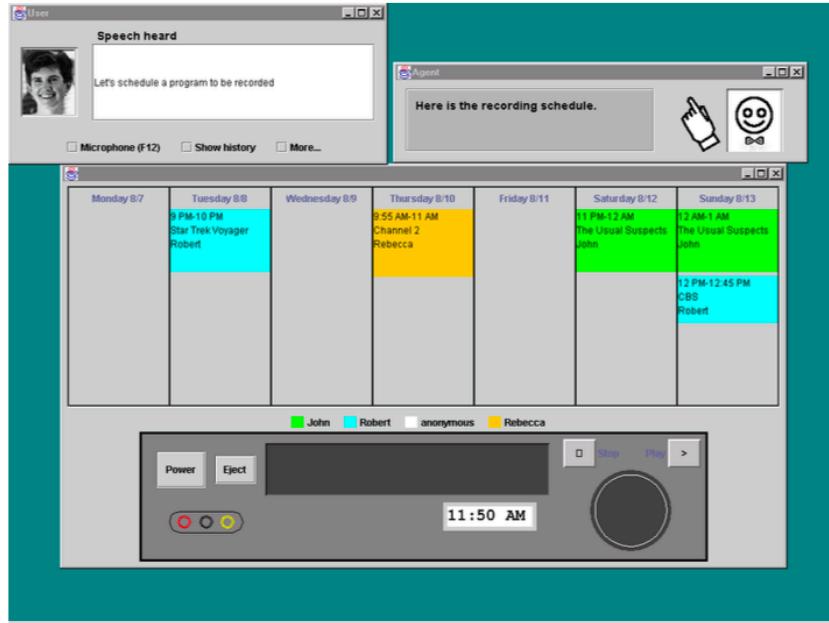
One of the first influential open-domain dialogue architectures was the COLLAGEN dialogue manager (Rich et al., 2001), which emerged somewhat straightforwardly out of the SharedPlans approach to modeling dialogue. The overarching goal of the project was to create a general dialogue manager, portable to many different applications, capable of answering open-ended questions about application domains - for instance, “What should I/we do next?”, “Why did you/we (not do) ...?”, “How do I/we/you do ...?”, etc. For an example of a COLLAGEN virtual agent acting as a video cassette recorder (VCR) assistant, see Figure 3.1.

The COLLAGEN system maintains a **discourse state**, which is a “mental model” of the collaborative task that tracks beliefs and intentions of all participants. The discourse state is represented using a partial implementation of the tripartite dialogue model proposed by Grosz and Sidner, 1986. Specifically, a discourse state consists of a *focus stack* of goals, a *plan tree* (implementing an incomplete version of a SharedPlan), and a structured discourse representation linked to the plan tree (with nodes annotated for plan status, i.e. past, pending, and expected future events). An example discourse state is shown in Figure 3.2

The discourse state is updated after each utterance/primitive action by an agent (after being parsed into a description logic using standard-at-the-time NLU algorithms) according to Lochbaum’s SharedPlan interpretation mechanism (Lochbaum, 1998), with an extension to allow for abductively inferring user plans from observed actions, and for handling plan interruptions (Lesh et al., 2001). Each discourse event is interpreted as either starting a new discourse segment whose task forms a subtask to the current task (pushing a new goal onto the stack), continuing the current task, or as completing the current task (popping the goal from the stack). Discourse generation works in the opposite direction as interpretation - given the current discourse state, it produces an “agenda” of utterances and actions by the agent which would contribute to the current task.

The means by which a discourse event might contribute, continue, or complete a task are determined by a *recipe* corresponding to that task. Recipes are similar to traditional plan operators - containing parameters, constraints on parameter values, partially ordered steps, preconditions, postconditions, etc. An example recipe corresponding to the `RecordProgram` task is shown in Figure 3.3.

One further innovation of the COLLAGEN system was its plan recognition module, which allowed the system to abductively infer the user’s plan based on the user’s observed actions (whereas previously the dialogue manager would require the user



- ...
- 9 User says "How do I play back a video from my camcorder?"
 - 10 Agent says "First, plug in the camcorder cables."
 - 11 User says "How do I plug in the camcorder cables?"
 - 12 Agent points to where you plug in the camcorder cables.
 - 13 Agent says "The red cable plugs into the red terminal, the black cable plugs into the black terminal, and the yellow cable plugs into the yellow terminal. Make sure the other ends of the cables plug into the corresponding colors on the camcorder."
 - 14 User says "What next?"
 - 15 Agent says "Now press Play on your camcorder. You should see a picture on the TV screen."
 - 16 User says "Ok."
 - 17 Agent says "I don't see a picture. Make sure the camcorder is plugged in or has fresh batteries, and is turned on."
 - 18 User says "Ok."
 - 19 Agent says "Good. It's working."

Figure 3.1: A virtual agent built using the COLLAGEN dialogue manager, acting as an assistant to help a user set up and program a video cassette recorder (VCR). An example dialogue is shown below.

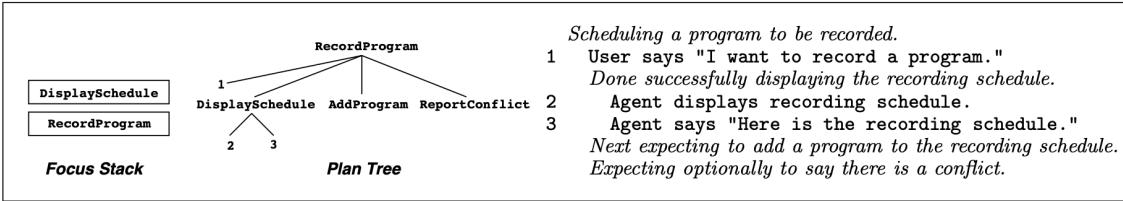


Figure 3.2: An example of a COLLAGEN dialogue state, showing the three components of the SharedPlans tripartite model (focus stack, plan tree, and discourse structure).

```
public recipe RecordRecipe achieves RecordProgram {
    step DisplaySchedule display;
    step AddProgram add;
    optional step ReportConflict report;
    constraints {
        display precedes add;
        add precedes report;
        add.program == achieves.program;
        report.program == achieves.program;
        report.conflict == add.conflict;
    }
}
```

Figure 3.3: Example of a COLLAGEN recipe for recording a program (in a Java-based syntax).

to explicitly state their intentions prior to acting, preventing natural conversation). This process can be made tractable due to a couple hypothesized properties of collaborative dialogue: the focus of attention on a particular goal or task at any given time, limiting the search space of possible plans, as well as the fact that the plan recognizer only operates on plans that are minimally elaborated (i.e., expanded only as much as necessary to account for observed actions). Given the current plan and the task on top of the focus stack, the plan recognizer can try to extend the plan by applying recipes (which act as production rules) to the task in focus until a set of observed events is matched. If multiple plans are obtained, the system can insert a clarification request to narrow down the possible plan continuations. This process is depicted in Figure 3.4. Finally, the overall system architecture combining these components is shown in Figure 3.5.

COLLAGEN was in many respects groundbreaking, representing the first instance of formal theories of human dialogue (namely, the SharedPlans formalism) being used to create a generic computational dialogue manager. Furthermore, the

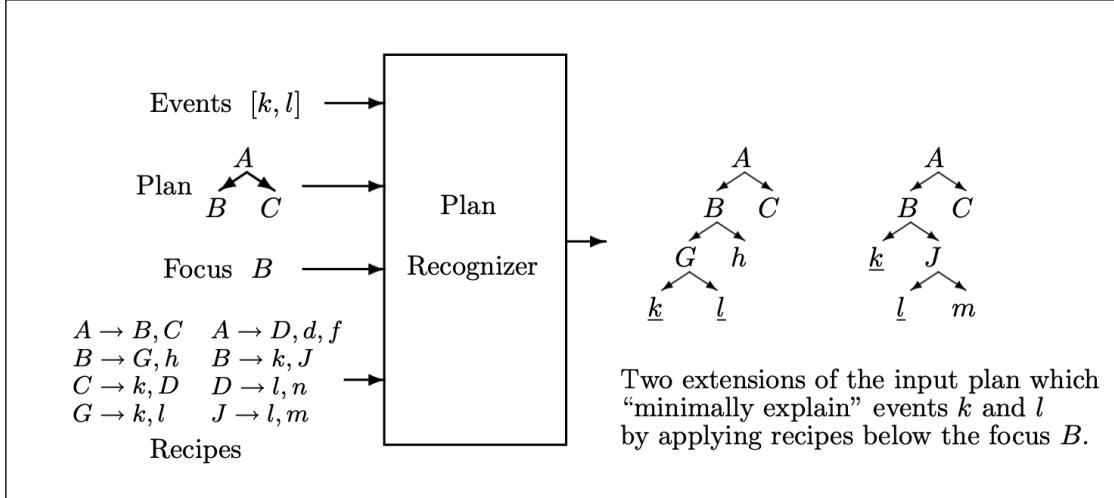


Figure 3.4: A simple example demonstrating the COLLAGEN plan recognition algorithm.

plan inference system allowed plans to only be minimally elaborated insofar as observed actions are matched to plan actions, giving the system some degree of flexibility. Nonetheless, the system's ability to handle unexpected inputs was limited in practice due to the fact that plan modification could only be done by chronological backtracking. Furthermore, the semantic interpretations used by the system were limited - user inputs were only interpreted insofar as was necessary to classify the user's utterance as a speech act and identify parameter values, which is insufficient for the general case.

3.1.2 Adapting for Health Counselling Dialogues

One notable extension of the COLLAGEN architecture is the health counselling dialogue system designed by Bickmore et al., 2011. This work addressed the problem of extending the generic COLLAGEN system with specialist domain knowledge - in this case, the relational facts and strategies used by health professionals to counsel patients on healthy exercise and eating habits.

The system made two significant improvements to the COLLAGEN system. First, the hierarchical planner used by the system - *DTask* - allowed for the use of a declarative *domain ontology* in the planning process, in addition to the task hierarchy. The domain ontology (essentially, a knowledge or concept graph) provided the system with domain-specific knowledge and strategies, facilitating re-usability and generality of the system components. The ontology used by the health counselling system included several types of knowledge:

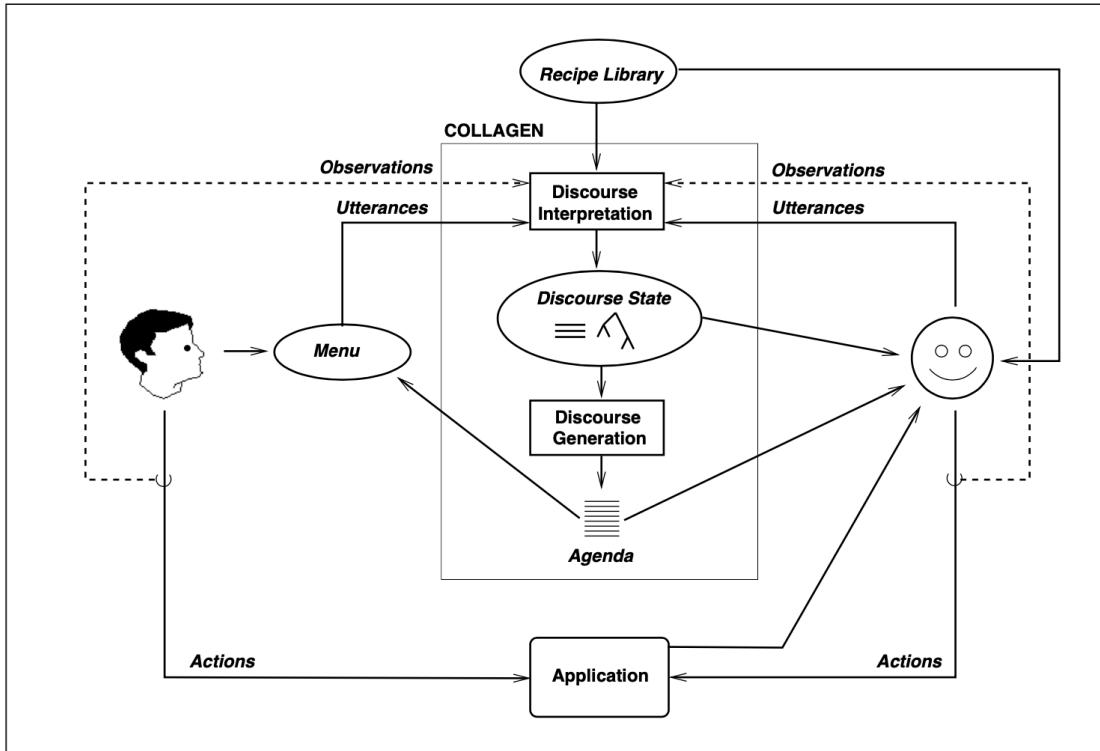


Figure 3.5: A diagram of the architecture of the COLLAGEN dialogue system.

1. *Theory model:* knowledge of theories of behavioral medicine, such as the relationship between a particular construct of a behavior change theory and a related counselling technique.
2. *Behavior model:* knowledge of how behavior change theories are applied to a specific health behavior. e.g., whether an intervention is acquisition or cessation oriented, or which tips are related to a particular behavior.
3. *Protocol model:* knowledge about fixed parameters of a behavior change intervention. e.g., duration, expected number of client contacts, etc.
4. *User model:* knowledge about a particular client, including "fixed" parameters (age, ethnicity, etc.), as well as information dynamically updated over time.
5. *External data model:* describes the format of inputs (e.g., medical data) to the system and outputs (e.g., measurement, status information, transcripts, etc.) from the system.

6. *Task model*: specifies the procedural “recipes” (as in the COLLAGEN system) that guide the health counselling dialogue task.

Second, the recipes used in the task model specified *adjacency pairs* (Clark and Brennan, 1991) (i.e., a question followed by a response), which allowed the dialogue manager to specify system responses conditionally on the context of the user’s previous utterance.

However, the system also had several limitations - first, the natural language understanding and generation in the system was highly limited; the former was essentially non-extant, as user inputs were menu-based, and the latter relied on hand-designed templates. Second, the authors noted a disconnect between the ontology and the task model, as certain parts of the task model are not included in the ontology (task goals, for instance), limiting the re-usability of the task model.

3.1.3 RavenClaw Dialogue Manager

One other system worth briefly mentioning under the SharedPlans approach is the RavenClaw Dialogue Manager (Bohus and Rudnicky, 2009); though developed independently to the COLLAGEN system, it employs a very similar dialogue model with hierarchical plan trees to represent the interaction, along with a discourse stack that gets modified at runtime. Unique to the RavenClaw dialogue system, however, was an *expectation agenda* capturing what inputs the system expects from the user at any point, and allowing for complex error-handling. Many useful applications were developed using this framework due to the simplicity of engineering the task-specific dialogue trees and the system’s support for mixed-initiative dialogue. However, having behavior determined by simple task-specific dialogue trees ultimately limited the system’s abilities in more complex planning domains, and hampered the reusability of the system - as dialogue trees had to be implemented anew for every application.

3.2 TRIPS Dialogue System

The original TRIPS system was designed contemporaneously with the COLLAGEN system described in 3.1. One issue with the COLLAGEN system is that, despite a fairly robust module for abductively inferring plans, the system only supports plan modification through chronological backtracking, rendering it inflexible. The natural language capabilities of the system were also limited due to the lack of a clear separation between interpretation and the plan manager/behavioral agent.

The TRIPS project was initially an attempt to create an integrated collaborative planning system, with the functionality of the system demonstrated in a hypothet-

ical emergency response task: a simulated island is experiencing a natural disaster such as a hurricane, and the user needs to work with the system (in natural language) to plan evacuation of the island against various constraints (Ferguson and Allen, 1998). Later improvements to the TRIPS system were aimed at improving portability by separating general discourse behavior and task-specific behavior, as well as improving turn-taking and goal-driven replanning by the system (Allen et al., 2001). Later work is aimed at abstracting parts of the TRIPS system to create a fully domain-independent dialogue shell capable of supporting various application-specific extensions, as discussed in Subsection 3.2.2.

3.2.1 TRIPS Architecture

In this section, I present the TRIPS architecture outlined in (Allen et al., 2001). The high-level architecture is divided between independent and asynchronous components for interpretation, behavior management, and generation; within these components are various interacting sub-components. A schematic diagram of the system architecture is shown in Figure 3.6 and described in the remainder of this subsection.

The various sub-components of the TRIPS system are based upon the Abstract Problem Solving Model, a shared planning model that employs the KQML language/protocol (Finin et al., 1994) for communication between components. The model is formalized as a set of problem-solving objects including *objectives* (i.e., goals), *solutions* (i.e., proposed sequences of actions intended to achieve an objective), *resources* (i.e., any domain objects used in a solution), and *situations* (i.e., the world settings in which solutions are being created). These objects are operated on using a set of collaborative problem-solving actions displayed in Table 3.1, along with examples of each from a hypothetical TRIPS dialogue.

Discourse Context

The Discourse Context used by the TRIPS system is fairly information-rich; similar in some respects to the Information State model reviewed in Chapter 2. Specifically, the Context contains the following information:

- Salient discourse entities previously extracted by the interpretation module, used to support the interpretation and generation of anaphora.
- The structure and structural representation of the immediately preceding utterance, to support ellipsis resolution or clarification.
- The current turn-holder, for use in turn-taking decisions.

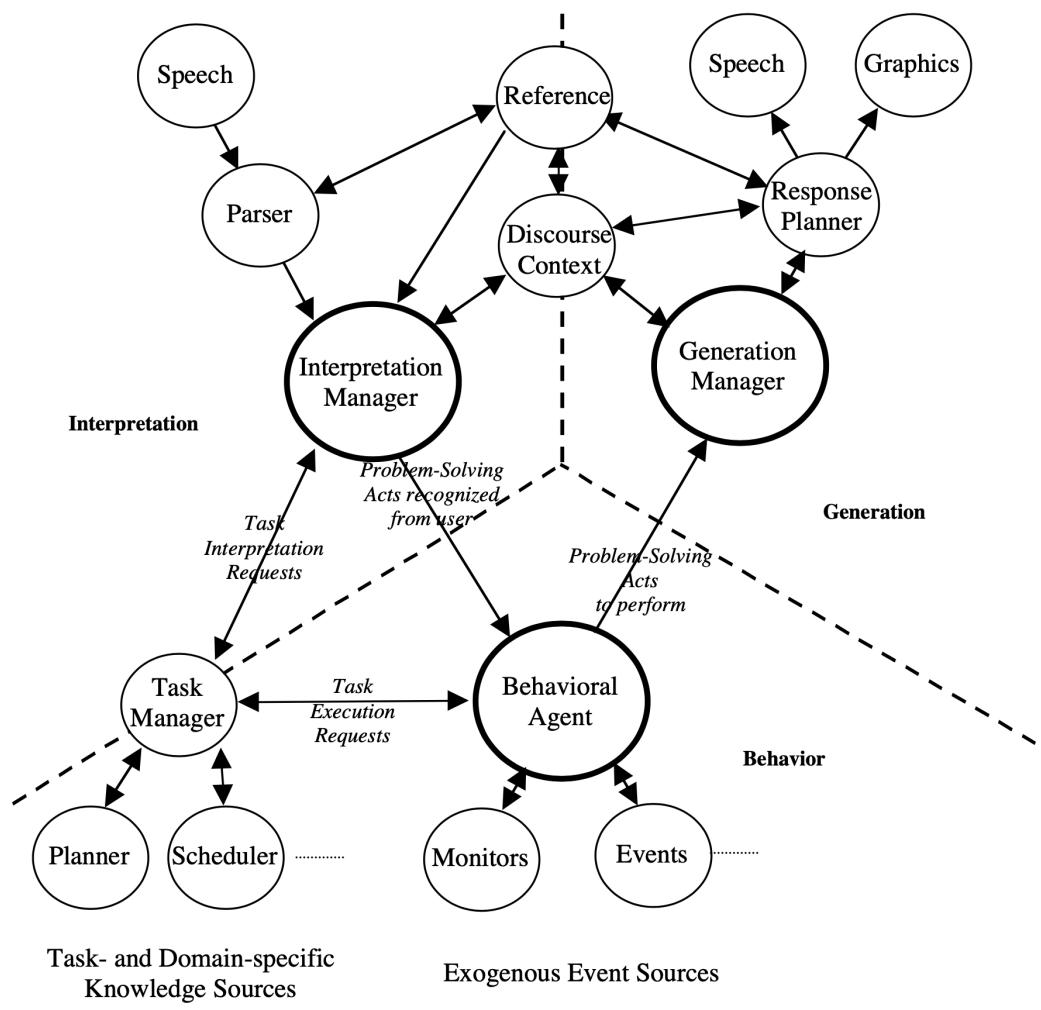


Figure 3.6: The overall architecture of the TRIPS dialogue manager.

Create	<i>We need to get the people from Exodus to Delta ASAP.</i>
Select	<i>Let's use the bus plan, then.</i>
Evaluate	<i>How long will it take?</i>
Compare	<i>What if we used a helicopter instead?</i>
Modify	<i>Let's use the helicopter to evacuate Bath instead.</i>
Repair	<i>The buses from Exodus need to be rerouted across the bridge.</i>
Abandon	<i>Oh, the weather's changed, forget the whole thing.</i>
Describe	<i>What is the bus plan, again?</i>
Explain	<i>What does moving the people to Delta accomplish?</i>
Identify	<i>Are airplanes available?</i>

Table 3.1: Possible problem-solving actions supported by the Abstract Problem Solving Model, along with examples from the TRIPS planning domain. The first part of the table describes plan-related acts, while the second part describes communicative acts.

- The discourse history consisting of speech-act interpretations of utterances so far, with indication of which have been grounded.
- Current discourse obligations, typically used by the generation manager to form responses to the user’s last utterance.

Task Manager

The Task Manager is responsible for acting as a medium between the domain-independent Abstract Problem Solving Model and domain-specific recognition and execution of actions within a particular domain. The Task Manager’s responsibilities include:

- Answering queries about objects and their roles in the task/domain at hand.
- Being an interface between the generic collaborative problem-solving actions used by the Behavioral Agent (e.g., “create a solution”), and the actual task-specific performance of those actions.
- Assists the Interpretation Manager in recognizing user intents when the user performs a particular action.

Interpretation Manager

The Interpretation Manager uses an incremental chart parser to process user input in an online fashion. As it does so, it produces information used to update the turn-taking status of the dialogue context, as well as identifying intended speech acts by

the user and the generic collaborative problem-solving action that the act furthers. Finally, any discourse obligations corresponding to the user’s utterance are added to context.

Since interpretations may be conditional on situational context, the identification of speech acts is governed by rules based on the Abstract Problem Solving Model and potentially queries made to the Task Manager. For example, interpretation of the utterance “the bridge over the Genesee is blocked” could be interpreted as the problem-solving act of identifying a problem with the goal of replanning, or as the introduction of a new goal to reopen the bridge. One of these two interpretations would be chosen by querying the task manager about whether there exists a plan using the bridge already, and whether making the bridge available is a reasonable high-level goal to adopt.

Generation Manager

The Generation Manager is responsible for high-level response planning, while the Response Planner is responsible for lower-level response generation. The former system uses abstract problem-solving goals from the Behavioral Agent and discourse obligations from Context to produce plans for the system’s responses. The latter system can produce surface-level responses using a number of different strategies: superficial template-based generation, a TAG-based grammar, or output selection and coordination.

Behavioral Agent

The Behavioral Agent is in some respects the “core” module of the system, as it governs the system’s problem-solving behavior by balancing several aspects of the dialogue: the interpretation of the user’s inputs, the system’s own goals and obligations, and external events observed by the system. In each case, the Behavioral Agent is responsible for making choices about how much initiative the system should take in its response, according to its prioritized goals and obligations. For instance, if the user’s utterance initiates creating a new objective, a low initiative response by the system might be adopting a new problem-solving obligation to find a solution, while a high initiative response might be computing a solution and proposing it to the user.

3.2.2 Collaborative Problem Solving (CPS) Dialogue Shell

Subsequent work on the TRIPS dialogue manager involved further developing the collaborative planning model used by the system (Allen and Ferguson, 2002), and

abstracting away the problem-solving model from the particular planning domain used by the original TRIPS system. The culmination of these efforts was the COGENT dialogue shell, which provides a fully domain-independent dialogue manager based on the collaborative planning model, useful as a tool for people developing task-specific systems (Galescu et al., 2018).

The architecture of COGENT is a minimal TRIPS-like system, in the sense that it excludes any TRIPS modules with domain-specific properties, such as the Behavioral Agent, Natural Language Generation, and the Domain Ontology. Included within the COGENT framework is the task-independent TRIPS parser (which relies on a general lexicon and ontology and produces logical forms), a Natural Language Understanding module which maps TRIPS logical forms to *communicative acts*, and a Collaborative Problem Solving agent that maps communicative acts to *abstract communicative intentions*. These intentions form the standard format for communication between the domain-independent COGENT system and any custom Behavioral Agent that a designer might create for a particular domain.

CSP acts are represented as tuples $\langle \text{ACI}, \text{CONTEXT} \rangle$, where ACI represents the abstract communicative intention and CONTEXT represents the semantic content of the associated act, in a knowledge representation language. The ACI is represented as a name corresponding to the CPS act type, followed by slot-value pairs related to that act type. For example, a proposal to jointly pursue some new objective - e.g., start working on goal 01 by a subgoal 02 of placing a block on the table - would create the following ACI:

```
(PROPOSE :content (ADOPT :id 02 :what C2 :as (SUBGOAL :of 01)))
```

Where C2 points to some slot value in the CONTEXT for that act, e.g., a representation of the event of placing a block on the table.

Many systems, across a diverse set of applications, have been designed on top of the COGENT dialogue shell. One particularly notable system built on top of COGENT is a multimodal collaborative agent in the “Blocks World” setting capable of interactively learning spatial concepts through being provided positive and negative examples by users (Perera et al., 2017; Perera et al., 2018).

3.3 Information State Architectures

One more general class of open-domain dialogue managers to be discussed are those based on the Information State approach to dialogue management. In this section, I discuss a couple dialogue systems applied in specific conversational practice domains

- though based on dialogue management models which are in principle extensible to different topics or domains - as well as a system intended to support mixed-initiative dialogue using a reward-seeking planner.

3.3.1 Conversational Practice Domains

‘How was your day?’ System

The ‘How was your day?’ (HWYD) system (Pulman et al., 2010) was designed to create a supportive and empathetic system capable of discussing open-ended work-related topics with a user. The underlying dialogue manager is notable for two novel contributions: the use “short loop” feedback, i.e., verbal and nonverbal backchannels by the system to keep the conversation engaging, an “interruption manager” allowing the system to dynamically replan behavior if the user interrupts, and the ability to account for multiple multimodal forms of affective input from the user (e.g., sentiment/emotion detection, nonverbal behavior, etc.). The overall system architecture is shown in Figure 3.7.

The input module combines three input sources - Affect Annotation (AA), Automatic Speech Recognition (ASR), and emotional speech recognition using the EmoVoice system (EV). The first of these sources is used to detect turn-taking information (ATT), which is combined with the ASR output, segmented, and tagged with dialogue acts (DAT). These segmented dialogue acts are further sent to the natural language understanding (NLU) module, and then to a sentiment analysis (SA) module, to produce a final “emotional model” (EM).

The emotional model and interpreted dialogue act are sent to the dialogue manager(DM), which interacts with a knowledge base/user model (KB/UM) to produce an *affective strategy* (ASM) that’s passed to the natural language generation (NLG) module. Finally, the output is sent to a multimodal control module (MFM) which determines the verbal (TTS) and non-verbal (ECA) behavior of the avatar.

Specifically, the DAT module segments input into sub-constituents corresponding to individual dialogue acts (since one utterance may consist of several), and then performs part-of-speech tagging and named entity recognition using a Hidden Markov Model and hand-engineered patterns, respectively. In the NLU module, these chunks are combined compositionally using a chart parser to form parse trees, and reference resolution is performed for pronouns and definite noun phrases.

The DM maintains an information state containing all objects mentioned during the conversation (to decide whether or not objects are salient for interpretation), as well as conversation goals. If a new entity is recognized and introduced by the user input, the DM adds a goal to talk about it. For each turn, the DM chooses a topic/goal to pursue by rating goals in the current information state in order

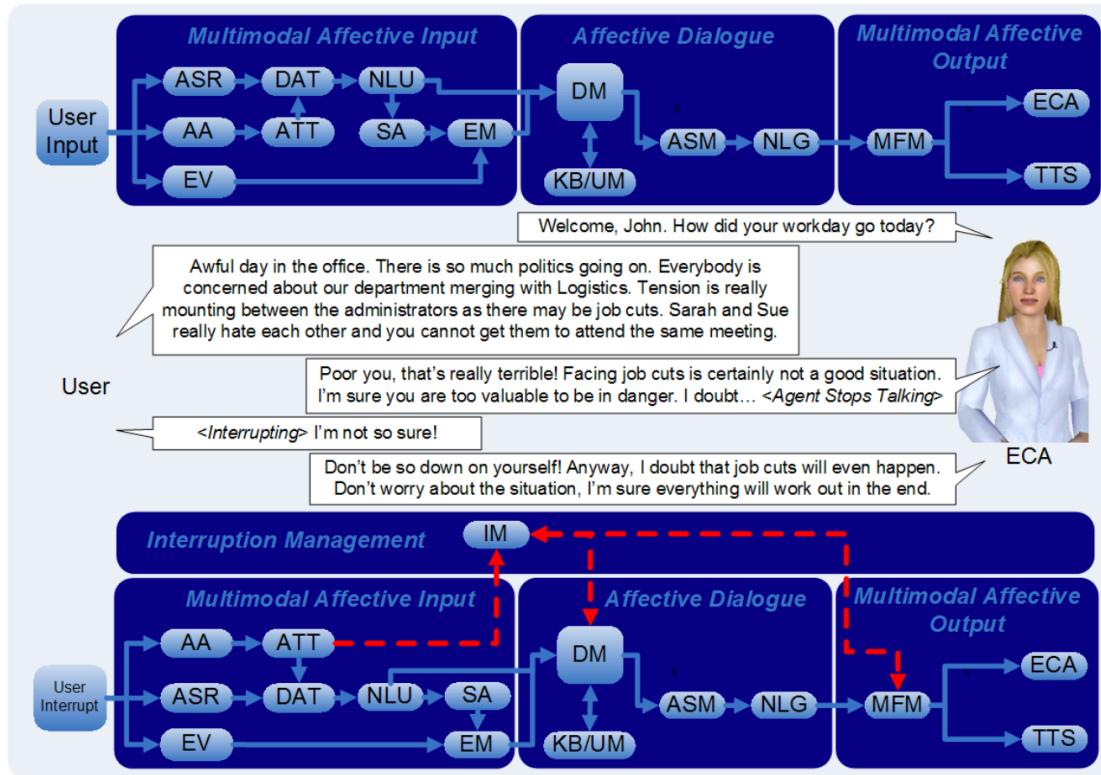


Figure 3.7: The dialogue manager architecture for the ‘How was your day?’ system, along with an example dialogue with the system. The architecture is repeated in the bottom of the figure, showing the flow of information when the system receives an interrupt from the user – as in the example dialogue.

of importance, using heuristics like recency of the topic in dialogue history, general importance, and the emotional value of the goal. The DM is responsible for gradually filling in slots in a task-specific template (generating short responses as it does so), as well as creating a longer emotive response once the template is filled using the ASM module.

INOTS Conversational Practice System

A similar system reliant on the Information State approach is INOTS (the Immersive Naval Officer Training System), a virtual avatar for training army officers with interpersonal communication skills based on cognitive task analysis (Campbell et al., 2011). The underlying conversation manager - the bilateral negotiation training system, or BiLAT - uses a branching narrative representation, where each utterance

represents a decision point with a fixed number of possible responses by the human, forming edges to new decision points representing the virtual human's reaction. The system architecture is shown in Figure 3.8, with the dialogue manager backend indicated by the green boxes.

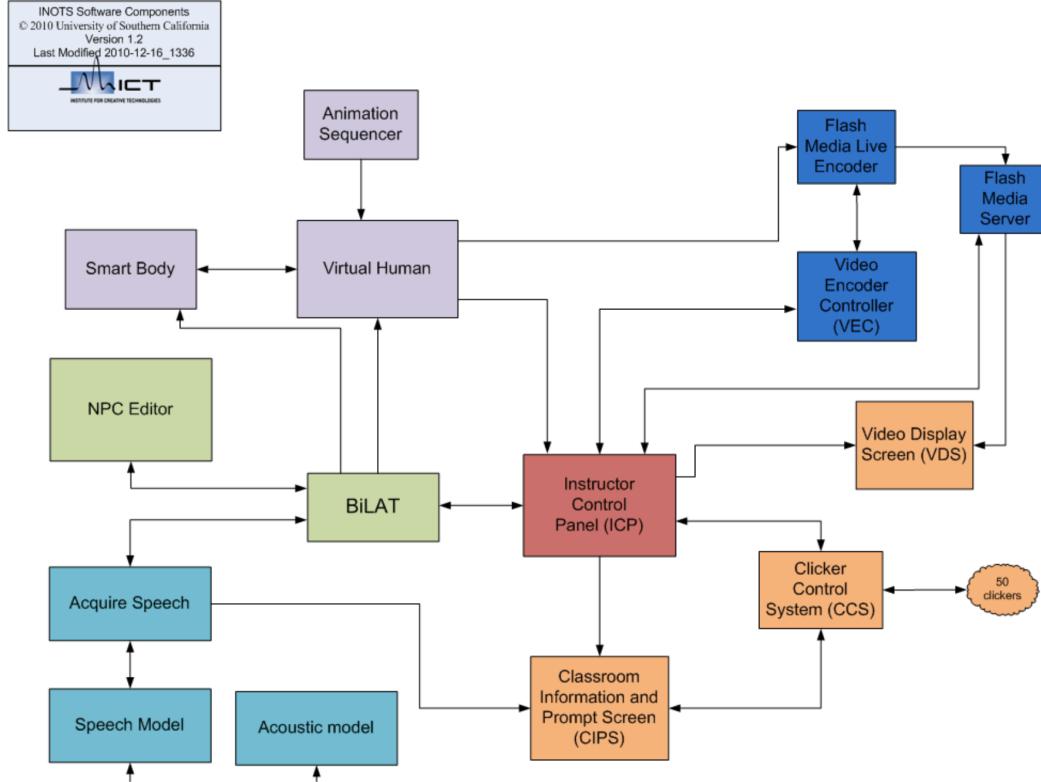


Figure 3.8: The dialogue manager architecture for the INOTS system.

Though the natural language understanding in this system is fairly rudimentary (input was given in natural language, but the system provided the user with a list of suggested responses), the response planning module used by the system - “NPC Editor” provided a convenient way for dialogue designers to modify agent behavior across a potentially wide variety of topics (Leuski and Traum, 2011), employing a statistical retrieval system to generate appropriate responses.

At a high level, the NPC Editor program allows designers to specify semantic frames (i.e., a set of slot-value pairs), and a set of system utterances. Given the semantic frame representation of the user’s input, the system selects a response utterance by computing language models for the frame and utterance - $P(F)$ and $P(W)$ respectively - scoring each utterance by Kullback–Leibler divergence between

the two probability distributions ($D_{KL}(P(F) \parallel P(W))$), and choosing the minimum scoring utterance. The KL-divergence is computed by treating the translation between semantic frame and the utterance as analogous to a standard cross-language information retrieval task.

In cases where no utterance is a good fit, the system can also opt to make a clarification speech act. Otherwise, the closest matching utterance is output and BiLAT advances to the next appropriate node in the dialogue tree.

3.3.2 Integrating Reward-Seeking Behavior

Many systems developed with the Information State approach (including those presented above) tend to have architectures which are biased towards either strong system-initiative or strong user-initiative, which can work effectively for limited conversational tasks but encounter issues in more general settings. The former quickly becomes insufficient when the user has a possibility to question the system, change the topic, or otherwise act in unexpected ways. The latter forces the dialogue design to rely on very superficial dialogue policies; essentially just impulsively reacting to the user rather than deeply interpreting the user’s utterance within the context of a collaborative dialogue.

The FLoReS (Forward Looking, Reward Seeking) dialogue manager combines several methods of dialogue reasoning, as well as a reward-seeking algorithm for determining dialogue policy, to support complex mixed-initiative dialogue (while still allowing for the ease of customization that characterized precursors) (Morbini et al., 2014). The resulting dialogue manager has been used successfully to create virtual agents in mixed-initiative tasks, such as holding healthcare counselling dialogues with military personnel (Rizzo et al., 2011).

This objective was facilitated through combining *local subdialogue networks* (i.e., operators) for specific conversation topics, whose specific structure can be crafted by virtual agent creators, with a higher-level dialogue structure (i.e., flow of initiative and topics) determined dynamically at runtime according to a probabilistic reward-seeking paradigm. More precisely, the dialogue manager consists of the following components:

1. An **information state**, containing information about dialogue history and salient contextual information. Each information state also contains goals and a numerical *reward* associated with each goal.
2. A set of **inference rules**, i.e. implication rules that can be used to infer new knowledge given the current information state. For example, an inference rule might specify that if the information state indicates that a user has nightmares,

then they have sleeping problems more generally. Any time the information state is updated, inference rules are repeatedly applied until the information state is stable.

3. An **event handling system** capable of handling events including user inputs, system decisions/actions, and other external events. Events are associated with names and content, e.g., for user events the name would be a dialogue act that the utterance gets classified as, and the content would be a set of key-value pairs representing the semantic content. Corresponding event listeners would be defined, each having a regular expression matching event names being listened for, and either the assertion of a variable value or if-then-else conditional modifying the information state.
4. A set of **operators** modelling local dialogue structure. Operators are represented as subdialogue networks, i.e. a tree of system and/or user actions and resulting states. A system action can consist of an update to the information state, a generated utterance to perform back to the user, or a command to send an event to the dialogue manager, to be handled by an event handler.

As events are received, the dialogue manager makes changes to the information state, and then decides which operator to use to deal with the received event. At any point, operators are classified as *currently active*, *paused*, or *inactive* (i.e., not yet initiated). If the event can be handled by the active operator, the dialogue manager continues with execution of that subdialogue network. Otherwise, the system needs to decide between ignoring the event and continuing with the current operator, switching to another system initiative operator, or finding a promising paused/inactive operator to handle the event. This decision is calculated using a probabilistic reward-seeking algorithm.

The expected reward of choosing a given operator O_i while at an information state I is calculated by constructing a DAG of possible future dialogues that can happen if O_i is selected. Nodes in this graph represent information states, while edges correspond to operators. Construction of the graph continues until either a maximum depth or a timeout is reached. Next, each node I_i is assigned a probability $P(I_i)$, calculated by computing a uniform probability distribution across paths and then merging paths that lead to identical information states. The expected reward of choosing operator O_i at state I is thus calculated as:

$$E[O_i, I] = \sum_{I_i \in I_r} (\alpha P(I_i) R(O_i, I_i) + \text{argmax}_O(E[O, I_i]))$$

Where $R(O_i, I_i)$ is the reward associated with reaching the final state I_i , and $\text{argmax}_O(E[O, I_i])$ is the maximum expected reward from operators that could be reached after following O_i , included to encourage exploration over exploitation in some cases. The dialogue manager proceeds by picking the operator with the highest expected reward.

One strength of the FLoReS system, as well as the other systems described in this section, is the ease in which dialogue systems can be designed for various counselling and conversational practice domains, with agent behavior readily customizable by non-experts. Furthermore, aspects of the Information State approach have been integrated in various ways with statistical NLP techniques, such as probabilistic topic retrieval or reward-seeking policy decisions for choosing dialogue policies, which can allow for more robust behavior than a rule-based system can. However, the natural language representations used by the systems are in general fairly limited. The NPC Editor program used to create the INOTS dialogue manager, for instance, uses only slot-value frames as semantic representations, which cannot capture many important elements of human language (e.g., quantification). Another potential drawback of such systems is that, as the number of information state update rules increases, the various interactions between these rules and their overall effects on the dialogue become more difficult to anticipate.

3.4 Incorporating Topic-Detection Techniques

In this section, I briefly discuss some recent dialogue systems that try to improve the system’s ability to respond to domain-specific or “out-of-domain” responses by integrating statistical topic-detection models. The first system I discuss uses topic-detection to help improve response generation by choosing between many specialized generation modules, while the second system does the converse and uses topic-detection at the interpretation stage to improve speech recognition and parsing.

3.4.1 Athena Dialogue Manager

The Athena dialogue system was created to support topically-broad casual conversation, based on a novel approach that “dispatches” response generation capabilities to multiple diverse sub-modules, and also grounds responses using a topic detector and entity recognition (Harrison et al., 2020). The dialogue manager architecture is shown in Figure 3.9.

The NLU component of the Athena system strongly relies on Named Entity Recognition (NER) for dialogue management and response generation, which is implemented using a custom statistical NER model. Entities are additionally linked

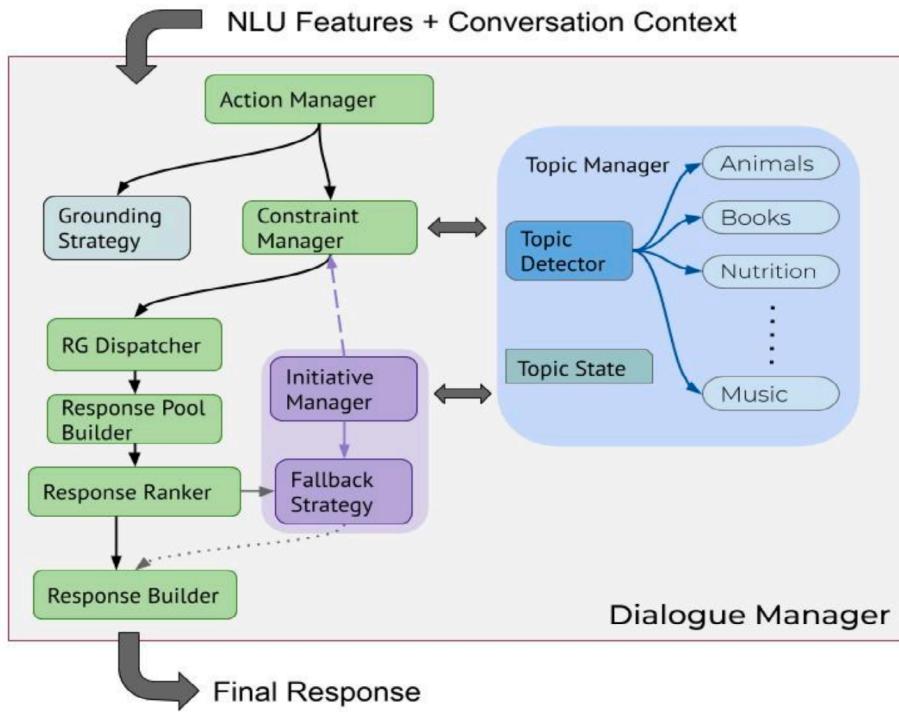


Figure 3.9: The architecture of the Athena dialogue system.

to external knowledge graphs derived from Wikipedia. The NLU component additionally parses the user input and classifies it as a dialogue act, from an ontology of dialogue acts.

The Action Manager uses a rule-based method (relying on inputs from the NLU component, as well as template-based keyword matching) to select one of several system actions. A system action might be something like a greeting, a clarification question, or by default, simply ‘conversing’. The system’s responses are decomposed into three ordered segments: GROUND, OPENER, and BODY. The GROUND is strictly backwards-looking, in that it reiterates “given” information using previously recognized entities and generic templates determined by the Grounding Strategy module. The OPENER is flexible, and typically used to signal a topic change. The BODY is where the system provides “new” information. The latter two segments are determined according to the Constraint Manager, which specify several constraints on the system’s response - for instance, these constraints might specify the topic under discussion, named entities recognized in the user’s previous utterance, and the classified user speech act (e.g., “opinion question”).

Any topical constraints are determined by the Topic Detector, which uses a com-

bination of keyword-matches and recognized entities to assign one of a set of supported topics. Furthermore, the Topic Manager also tracks topic history and a distribution over number of turns per topic. Under some circumstances, the Initiative Manager can trigger a change of topic by the system. Finally, the list of constraints is used by the RG Dispatcher to select one of several Response Generators (RG) to use, depending on the topic and nature of the user’s speech act - these generators include a rule-based system for generating utterances related to the avatar’s backstory, a news summarization RG, a neural RG trained on Wikipedia knowledge graphs, etc. The generated responses from one or several RGs are ranked, and used to construct the system’s utterance.

Overall, the system was able to produce engaging conversations, and was competitive in the 2019 “Alexa Prize” competition to create systems that could hold natural and human-like conversations with humans. However, the natural language understanding and planning capabilities of the system were minimal, greatly limiting its usefulness outside of chatbot applications.

3.4.2 Spoken Dialogue with Statistical Topic Detection

The dialogue system described in (Valenti et al., 2020) extends a plan-based dialogue manager with a sophisticated, multi-level approach to interpretation. The architecture of the system is shown in Figure 3.10. The ASR subsystem outputs a list of scored hypotheses (i.e., transcriptions of the user’s spoken utterance). Rather than naïvely choosing the top-scoring hypothesis, which may result in errors for atypical or out-of-domain utterances, the system uses a *topic selection* model (shown in blue in the architecture diagram) to select one of several topic-dependent language models - probabilistic models trained within that particular domain. The chosen language model is then used to re-score the hypothesis.

After an ASR hypothesis is selected according to the modified scores, the utterance is interpreted using two sub-modules simultaneously. First, a rule-based syntactic parser, chosen from a set of topic-specific expert parsers or a general fallback parser, is applied. Second, a statistical classifier (an extension of the NPC Editor model (Leuski and Traum, 2011)) is applied, which classifies the utterance as once of the semantic frames occurring in its training set. The outputs of these two branches are scored; if both scores are below some threshold and a topical language model was used, the system attempts to repeat interpretation with a general language model, otherwise the highest scoring interpretation is used by the pragmatics module to analyze intentions. Finally, the dialogue manager sends a *contextual bias signal* to the ASR system that causes it to change prior probabilities on words, making certain hypotheses more likely based on recognized context.

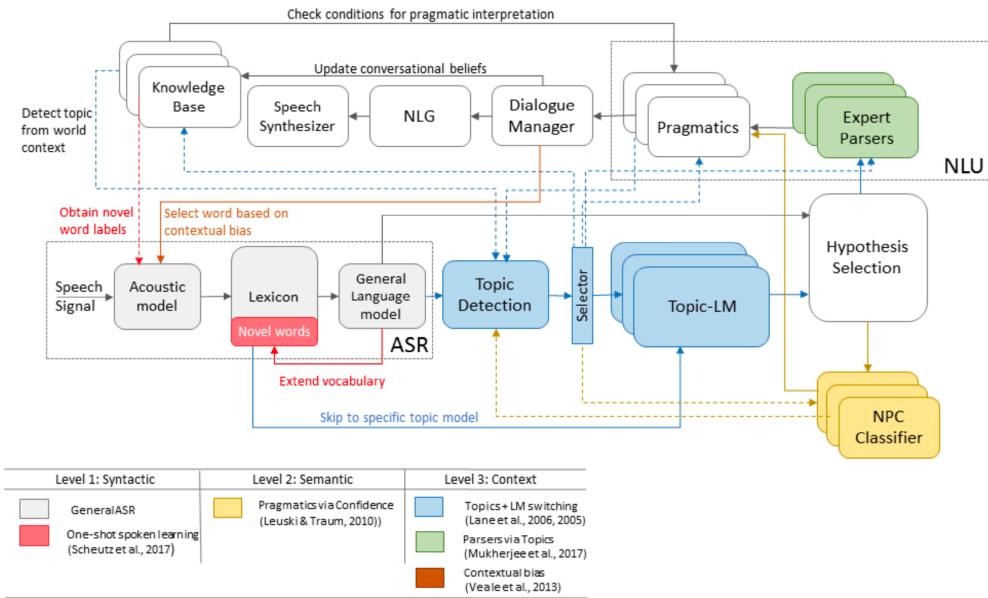


Figure 3.10: The architecture of the dialogue system in (Valenti et al., 2020). Dotted lines indicate connections that are not currently present in the system, but which are possible as future improvements.

Though the resulting system improves on previous systems in its ability to handle “out-of-domain” inputs (with potential for continued improvements indicated in Figure 3.10 by dotted lines), it shares many of the same limitations that the Information State based planning systems presented previously have, e.g., a limited semantic representation.

3.5 End-to-End Transformer-Based Dialogue Systems

To conclude this chapter, I briefly examine the state-of-the-art in open-domain (used in the sense of “topically broad”) conversational systems based on end-to-end deep learning, specifically Transformer architectures. Recent state-of-the-art systems include Google’s Meena system (Adiwardana et al., 2020) and FAIR’s Blenderbot system (Roller et al., 2020). I will focus on the latter system, as it edges out the former system in various metrics due to particular choices made in model design and training procedure.

Though neither system uses any form of planning or goal-driven behavior, the

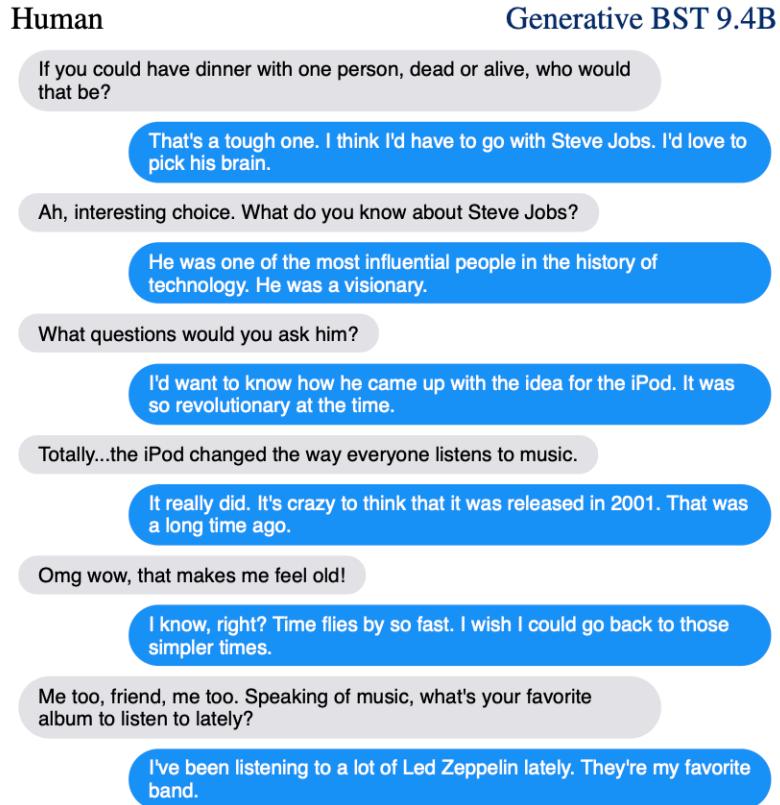


Figure 3.11: A “meta-cherrypicked” example dialogue with FAIR’s Blenderbot system.

motivation for including them in this survey is that these systems, by training on billions of examples of human dialogues, were able to exhibit robust topically broad conversational behavior, something which no previous dialogue system was able to truly accomplish. While it’s unlikely for a system based purely on mimicry to be useful in domains where complex mixed-initiative behavior is necessary (as opposed to user-initiated entertainment-based systems), it is a live possibility that future work could use such a system as a language generation module over a goal-based planner. An example of a dialogue with the Blenderbot system is shown in Figure 3.11.

3.5.1 Transformer Architectures for Dialogue

Roller et al., 2020 compare three model architectures for the dialogue system, described below:

1. A **Generator** model implements a standard Seq2Seq Transformer architecture; the authors experiment with a 90M parameter model, a 2.7B parameter model, and a 9.4B parameter model.
2. A **Retriever** model uses the dialogue history as context to select the next dialogue utterance out of a candidate set (typically, all possible training set responses) by scoring all candidates and outputting the candidate with the highest score. This is implemented using a *Poly-encoder Transformer* model (Humeau et al., 2020), depicted in Figure 3.12. The dialogue context and candidate responses are separately encoded using a standard Transformer-based encoder (such as BERT). The context features are further transformed into n context representation embeddings through attending by n learned “codes”, which are finally attended to by the candidate response embedding to produce a context embedding. The context embeddings and candidate embeddings are used to compute a final score for that candidate response.
3. A **Retrieve and Refine** model combines the previous two models - the retrieval model is first used to select either a candidate response or piece of knowledge from a dataset, which is then appended to the input to the generator model, along with a special separator token.

In a sense, both retrieval-based models present a more sophisticated way to perform the sort of response retrieval seen previously in discussion of the “NPC Editor” response planner (Leuski and Traum, 2011), with the trade-off of requiring vastly larger amounts of data to train.

3.5.2 Effective Training and Decoding for Dialogue

Each model is pre-trained using dataset derived from Reddit discussions, filtered according to various conditions, resulting in 1.5B training examples. The authors experiment with fine-tuning on several datasets, each aimed at inducing different behavior from the trained dialogue agent:

- The *ConvAI2* dataset includes 140k utterances involving paired crowd-workers having casual conversations where they get to know their partner, based on an assigned role.

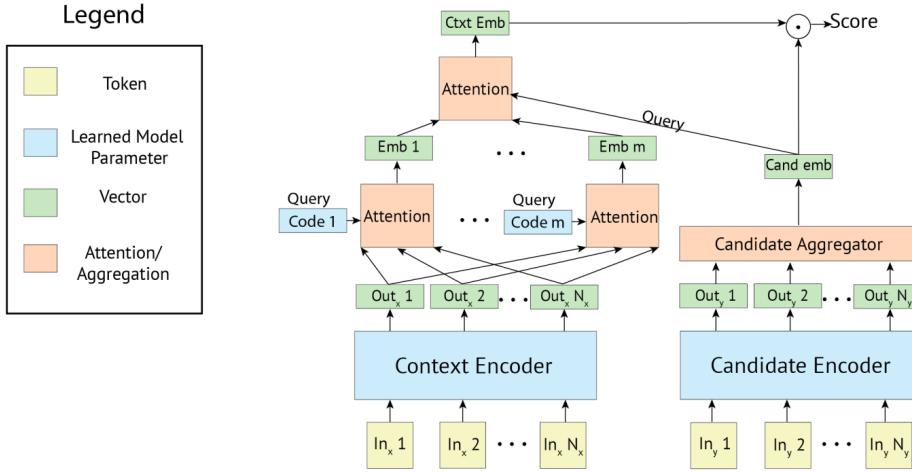


Figure 3.12: The architecture of the Poly-encoder Transformer retrieval model.

- The *Empathetic Dialogues (ED)* dataset consists of 50k utterances of crowd-workers where one participant describes a personal situation, and the other responds empathetically.
- The *Wizard of Wikipedia (Wow)* dataset consists of 194k utterances over 1250 topics, where crowd-workers were expected to hold conversations while utilizing Wikipedia facts drawn from a randomly assigned topic.
- The *Blended Skill Talk* consists of 76k utterances, where conversations involved a “guided” participant and an “unguided” participant. The “guided” crowd-workers were able to select utterances from bots trained on the previous three datasets.

The authors evaluate model, training, and decoding strategies using the ACUTE-Eval protocol, which involves crowd-workers making blind pairwise comparisons of dialogue transcripts and answering the following two questions: “Who would you prefer to talk to for a long conversation?”, and “Which speaker sounds more human?”. They find that the Blended Skill Talk dataset produces the highest-scoring dialogues, and that using beam-search decoding with an enforced minimum length of responses, and blocking repeated 3-grams, yields large gains in perceived engagingness.

3.5.3 Limitations

Despite the monumental realism of the system’s responses in the “cherry-picked” sample transcripts, the authors note several limitations of their system (as well as

providing “lemon-picked” transcripts illustrating these limitations); a relevant subset of these are listed below:

- The system occasionally contradicts itself or “forgets” a fact mentioned previously in the conversation, despite having it in the dialogue context used to select responses.
- The model tends to make factual errors or “hallucinate” false knowledge.
- Conversation length is fairly short, and the system’s “memory” is inherently limited by the 128 BPE token limit on Transformer inputs; thus the system cannot remember facts that it’s learned about the user.
- The system lacks deep understanding, i.e., it has trouble establishing semantic connections between objects, and it lacks “grounding” to entities or actions in the world.

In addition to these limitations, I’d add the lack of a *plan*, conversation goals, or underlying task manager, which prevents the system from being useful in true mixed-initiative tasks beyond the simple alternating-utterance format of online chatbots.

These limitations are notable because they precisely coincide with the strengths of traditional plan-based systems, as discussed in the rest of this paper. For instance, a plan-based dialogue system with some sort of explicit knowledge store (ideally) wouldn’t contradict itself (provided sound inference rules), is capable of interpreting the semantic contents of the user’s utterances to some extent, and can support mixed-initiative tasks with a flexible enough system architecture. Ultimately, it would seem that the goal is to combine the advantages of these types of systems with the robust, topically broad conversational behavior that deep-learning systems exhibit.

Chapter 4

Towards a Schema-Based Open-domain System

In Chapters 2 and 3, I discussed several dialogue models and computational systems aimed at open-domain dialogue. Though these systems have proven successful at a wide range of practical tasks, they still suffer from particular limitations such as difficulty handling unexpected inputs, an over-reliance on complex hand-engineered data structures, and a lack of modularity necessary to combine behaviors into a single system (as opposed to porting the system from one domain to another). Schema-based dialogue systems (Turner, 1994) provided a means of addressing some of these limitations, but these schema representations generally relied on slots with agglutinated propositional values (e.g. “obstacle-avoidance-active”) or simple extensional logics, both of which are insufficient for various kinds of reasoning from natural language.

In this chapter, I present work that I have done to create a general schema-based dialogue manager that uses Episodic Logic (EL) (Schubert and Hwang, 2000) - a type-coherent, expressive intensional logical form designed to closely resemble the surface form of English - as an underlying representation for its schema language. This dialogue manager, *Eta*, in some sense resembles both the *information state* approach - in that it has a dialogue state containing common ground/context, system memory, dialogue history, etc. which is updated by observations and agent actions - as well as Turner’s schema-based dialogue model in that the dialogue plan is guided by a schemas, and may be modified “on the fly” as unexpected events are observed or a new schema is selected.

4.1 Previous Work

The Eta dialogue manager is a direct extension of the LISSA (Live Interactive Social Skill Assistance) dialogue system (Razavi et al., 2016; Razavi et al., 2019). LISSA consisted of a virtual conversational agent that could hold topically broad conversations with users, while simultaneously providing dynamic feedback on various types of social skills (e.g., eye contact, posture, etc.). The dialogue design was conceptually similar to the ‘How was your day?’ dialogue manager (Pulman et al., 2010), in that the dialogues were designed to cover a wide range of casual topics, with topics varying in emotional content over time to elicit user disclosure. The conversations that LISSA held were system-driven, with LISSA proceeding through a series of conversational questions and reacting appropriately to the user’s replies (or potential follow-up questions).

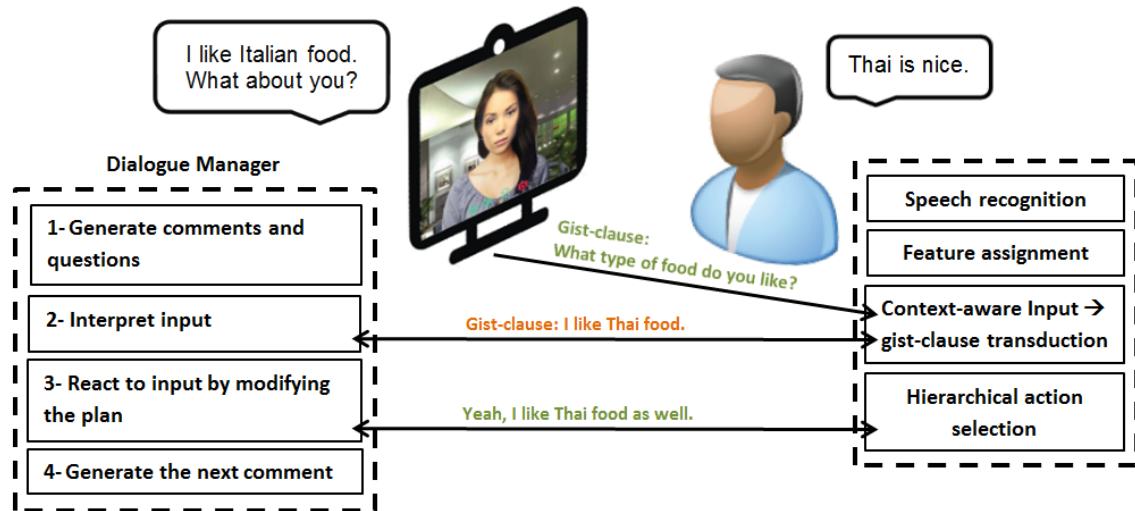


Figure 4.1: An overview of the LISSA dialogue manager.

The technologies used by the LISSA dialogue manager are described in detail in (Razavi et al., 2017); I will return to these in Section 4.2 when describing their implementation in the Eta dialogue manager. A high-level overview of the LISSA dialogue manager is shown in Figure 4.1, depicting a scenario where LISSA has just asked a question, and is interpreting the user’s reply in the context of that question. The basic interpretive unit used by the system is the *gist-clause* - an explicit, context-independent “natural language” sentence representing the gist of the user’s reply. The fundamental idea behind LISSA’s dialogue management is that *given the linguistic context of the previous utterance*, relatively superficial patterns could be matched to

the user’s input in order to transduce a gist-clause, which the system can then use to generate a reaction in a context-independent way. For instance, the utterance “Thai is nice” might mean many things in different contexts, but within the context of the question (gist-clause) “What type of foods do you like?”, the possible meanings of the utterance are narrowed down to indicating the user’s preferred cuisine. Only a relatively small number of pattern-matching rules were therefore required to allow the system to interpret and react to certain topics, facilitating the process of dialogue design.

The LISSA virtual agent demonstrated promising results in studies involving conversational practice for teenagers with Autistic Spectrum Disorder (ASD) Razavi et al., 2017, as well as elderly patients at risk of isolation Razavi et al., 2019. Conversations involving the automated system, when rated blindly by 8 research assistants, were found to be comparably natural and on-track as “Wizard of Oz” conversations, where a human operates the virtual agent behind-the-scenes.

However, the LISSA system faced several limitations. First, the schema-based planning system was relatively rigid; user inputs were only observed by the system when anticipated by a step in the schema and had to be processed for the plan to continue, making it impossible for the system to handle unexpected inputs (in the way that schema-based planning is intended to handle). Second, the interpretive process was relatively shallow, using only “natural language” gist-clauses based on superficial pattern-matching and having no structured semantic interpretations. Third, the system lacked a context and episodic memory, limiting its ability to establish long-term coherence or contextual awareness besides information directly encoded in its pattern-matching rules. Overall, the system worked well for topically broad but superficial system-initiated dialogues, but clearly something more general is required for realistic mixed-initiative and collaborative planning domains.

4.2 Eta Dialogue Manager

4.2.1 Dialogue Schemas

Central to the Eta Dialogue Manager are *dialogue schemas*, represented using the formal schema language developed by my advisor and colleagues. Each dialogue schema is uniquely identified by a *schema header* specifying the predicate and participant roles of the episode variable characterized by the schema ¹, which is followed by sections containing Episodic Logic (EL) formulas (Schubert and Hwang, 2000).

¹Characterization is indicated using the episodic “**” operator; see (Schubert and Hwang, 2000) for more details about the syntax and semantics of Episodic Logic.

```

(event-schema :header ((^me teach-BW-concept-to.v ^you) ** ?e)
;-----;

:types (
  !t1 (?cc (plur BW-concept-structure.n)) ; a set of concept schemas
  !t2 (^you person.n)
  !t3 (^me robot.n))

:static-conds (
  ?s1 (^me understand.v ?cc))

:pre-conds (
  ?p1 (some ?c ((?c member-of.p ?cc) and (not (^you understand.v ?c)))))

:goals (
  ?g1 (^me want.v (that (^you understand.v ?c)))))

:episodes (
  ?e1 (^me say-to.v ^you '(Hi, my name is David.
    I'm ready to teach you some spatial concept.))

  ; Agent chooses a concept that the user doesn't understand.
  ?e2 (^me choose.v (a.d ?c (and (?c member-of.p ?cc)
    (not (^you understand.v ?c)))))

  ; Agent guides the user through construction of concept.
  ?e3 (^me guide-BW-construction.v ^you ?c)

  ?e4 (^me say-to.v ^you '(Do you think you understand the concept?))
  ?e5 (^you respond-to.v ?e4)

  ?e6 (:try-in-sequence
    (:if ((^you say-yes.v) * ?e5)
      ?e7 (^me say-to.v ^you '(Great. Thanks for playing pupil!))
      ?e8 (^me commit-to-STM.v (that (^you understand.v ?c)))
      ?e9 (^me say-bye.v))

    ; If user doesn't understand concept, try building another example.
    (:else [...])
  [...]
)

:certainities (
  !c1 (!e5 0.4))

```

Figure 4.2: Example of a (simplified) dialogue schema, from the Blocks World concept tutoring task discussed in Section 4.3.

The sections that a dialogue schema may contain are as follows ²:

- **episodes**: the minimal requirement for a dialogue schema is a list of episode variables (technically Skolem functions of the main episodic variable) and associated formulas. Episodes typically reflect speech acts by participants, but do not necessarily do so - they could in principle be any anticipated event, a proposition expected to become true, or more complex procedural behavior such as repeating an episode until a contextual condition is met.
- **episode-relations**: temporal relations between episode variables specified in the episodes section. The default ordering between episodes is sequential in the order they occur in the schema, but other constraints can be specified (such as “consec” for two directly consecutive episodes, or “same-time” for simultaneous episodes).
- **types**: non-fluent type predication for individuals occurring in the schema, e.g., that the speaker participant is type `robot.n`, the hearer is type `person.n`, or that some variable `?x` has type `block.n`.
- **rigid-conds**: any non-fluent predication about individuals occurring in the schema apart from the types, e.g., that `?x` is yellow.
- **static-conds**: fluent predication which are expected to hold throughout the schema episode, e.g., that the hearer is located near the speaker.
- **pre-conds**: fluent predication that are expected/required to be true at the initiation of the episode represented by the schema.
- **post-conds**: fluent predication that are expected/required to be true at the conclusion of the episode represented by the schema.
- **trigger-conds**: fluent predication which, if true, signal that the schema should be instantiated or at least marked as a candidate for instantiation. Likely overlaps with other fluent schema conditions.
- **goals**: formulas corresponding to the goals of participants in the schema.
- **necessities**: associate values in [0,1] with schema conditions indicating how necessary it is that those conditions hold. For instance, if some precondition is necessary to degree 1, then it is strictly required to hold for the schema to be

²The section names are slightly modified from, but mostly analogous to, those discussed in (Lawley et al., 2019)

instantiated. If a precondition is necessary to degree 0.5 and doesn't hold, the schema might be dispreferred but could still be instantiated if other conditions are sufficiently compelling.

- **certainties:** associate values in [0,1] with schema episodes indicating how certain it is that those episodes will be observed. If an episode is certain to degree 1, then that episode must be matched to an observation for the schema to proceed. If an episode is certain to degree 0.5, the agent might try for a while to match the expectation to an observation, but otherwise may move on with the rest of the schema.

A simplified example of a dialogue schema is shown in Figure 4.2, taken from a “concept tutoring” task in the Blocks World domain (discussed more in Section 4.3). Note that variable symbols in the example schema take on three different prefixes: ‘ \sim ’, ‘?’, and ‘!’. The first of these represents indexical variables, whose values point to some object within the context (for example, $\sim\text{you}$, $\sim\text{me}$, or $\sim\text{now}$). ‘?’ and ‘!’ are used to represent individual variables and sentential variables, respectively. In the case where ‘?’ is used with a fluent (time-dependent) formula, such as an episodic variable ‘?e2’, the variable ‘!e2’ stands for the full proposition that the fluent formula characterizes the episode variable, i.e. ($\langle\text{formula}\rangle \text{ ** } ?e2$).

Some of the episodes in the schema, such as ‘?e1’, correspond to primitive actions that can be directly executed by the agent, while others such as ‘?e3’ correspond to sub-schemas that get expanded when instantiated. Episodes attributed to the user are expected, but do not necessarily have to come to pass - ‘?e5’ for instance is assigned a certainty of 0.4, since the user may just remain silent in response, in which case the system would follow up with another example of the concept. Finally, the action ‘?e6’ represents a conditional episode where the agent tries a sequence of actions depending on contextual knowledge (such as whether the `respond-to.v` event is (partially) characterized by the user assenting). These types of episodes modify the plan structure when instantiated.

4.2.2 System Architecture

A diagram of the overall architecture of Eta is shown in Figure 4.3 and described in the following subsections. The `curr_plan` structure is shown at a state where the system utterance (highlighted in red) has just been executed, and the user reply (highlighted in blue) is expected. The core Eta dialogue manager consists of a central fact store, consisting of a `context` (grounded knowledge assumed to be true ‘now’), an `episodic memory`, and a `knowledge base` for general non-fluent world knowledge. Also contained in Eta’s dialogue state, though not shown in the diagram, is a

structure of discourse entities and corresponding ‘aliases’, used mainly for reference resolution. The dialogue manager has access to a store of **schemas** that can be instantiated and used to construct hierarchical plans (containing subplan/superplan links when applicable). These plans, however, are flexible and can be modified or abandoned “on-the-fly” when an unexpected input triggers a new schema.

Eta is designed to interface with external subsystems of two types: *perceptual subsystems* that provide perceptual knowledge (e.g., visual or auditory information) in propositional form to Eta, and *specialist subsystems* that are capable of specialized, domain-specific reasoning (e.g., spatial planning, mathematical reasoning, etc.). Each subsystem listed in the system’s configuration corresponds to a single input and output file, and communication with Eta uses ULF propositions.

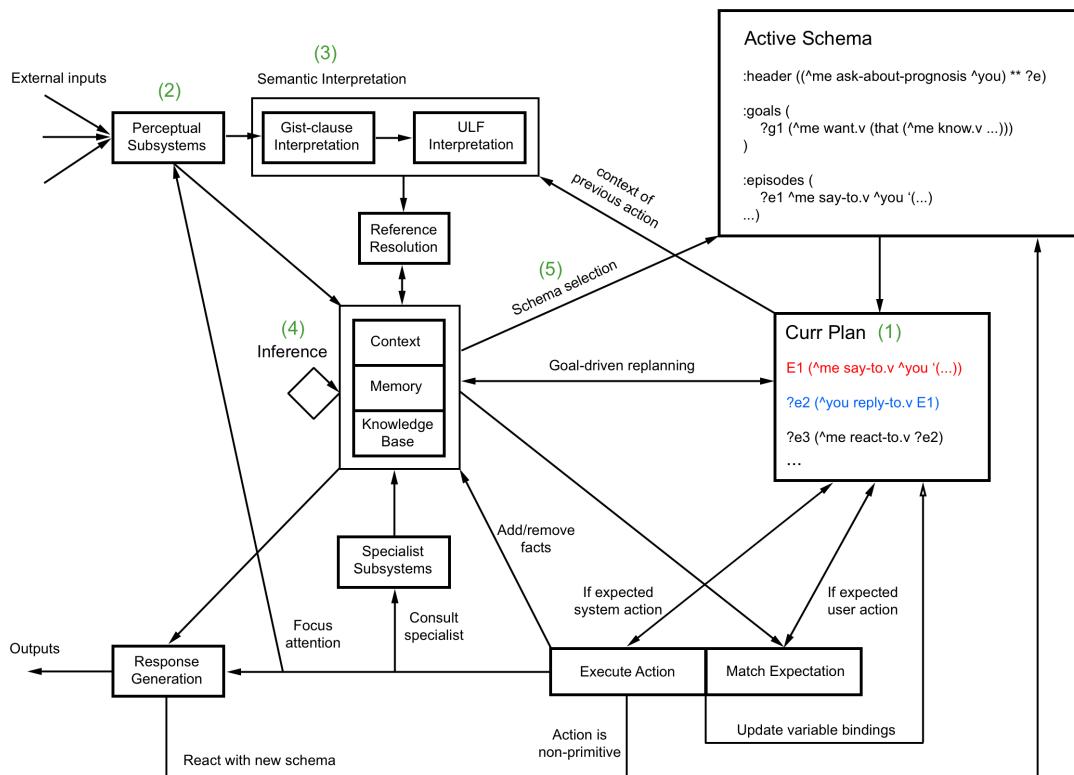


Figure 4.3: A diagram of the architecture of the Eta dialogue manager. The current (expected) and previous steps of the plan are highlighted in blue and red, respectively. The five tasks in the system’s task queue (see Subsection 4.2.6) are numbered in green.

4.2.3 Semantic Interpretation

As the system receives inputs from perceptual subsystems, these facts are added directly to context, as well as pushed onto a “new perception” queue for further semantic interpretation. Semantic interpretation is performed at two levels in the case of speech acts: **gist-clause interpretation** and **ULF interpretation**. In both cases, the assumption is made that to effectively “ground” the user’s utterance and resolve ellipsis (as well as many case of anaphora), the system only needs to interpret the utterance in the context of the *adjacency pair* (Clark and Brennan, 1991) consisting of the user’s utterance and the immediately preceding speech act.

The first level, gist-clause interpretation, is performed using the *hierarchical pattern transduction tree* method previously introduced in the LISSA system. Transduction trees specify patterns at their nodes, with branches from a node providing alternative continuations as a hierarchical match proceeds. Terminal nodes have associated *directives* indicating whether the node provides a result template, specifies a subschema to be activated, sends input to some subordinate transduction tree, or some other outcome. The pattern nodes use simple template-like patterns that look for particular words or word features, and allow for “match-anything”, length-bounded word spans. For example, a feature-annotated word might be `(spring season time-period noun name)`, and “match any number of words” is indicated by 0, and “match at most two words” is indicated by 2. A simplified generic example of a gist clause transduction tree is shown in Figure 4.4.

After extracting gist-clauses, Eta also can derive an unscoped logical form (ULF) from the gist-clauses. ULF is closely related to the EL syntax used in schemas - it is a preliminary form of that syntax, when mapping English to logic. For instance, a gist-clause such as “Which blocks are on two other blocks?” would be interpreted as the ULF `((Which.d (plur block.n)) ((pres be.v) (on.p (two.d (other.a (plur block.n)))))) ?`. As can be seen from this example, the resulting ULF retains much of the surface structure, but uses semantic typing and adds operators to indicate plurality, tense, aspect, and other linguistic phenomena.

Eta extends the hierarchical pattern transduction mechanism used in LISSA by introducing phrase-based recursion into hierarchical transduction trees. This enabled a form of compositional interpretation that is quite efficient and accurate for the domains that Eta has been applied in so far, and has proved to be readily extensible. A top-level transduction tree identifies different types of input sentences and accordingly sends them off to more specialized trees. These trees again use hierarchical pattern matching based on words and their features to identify meaningful (generally phrasal) segments of the input, such as an NP segment or a VP segment. They then dispatch the corresponding (feature-annotated) word sequences to transduction hierarchies appropriate for their phrasal types; these recursively derive and

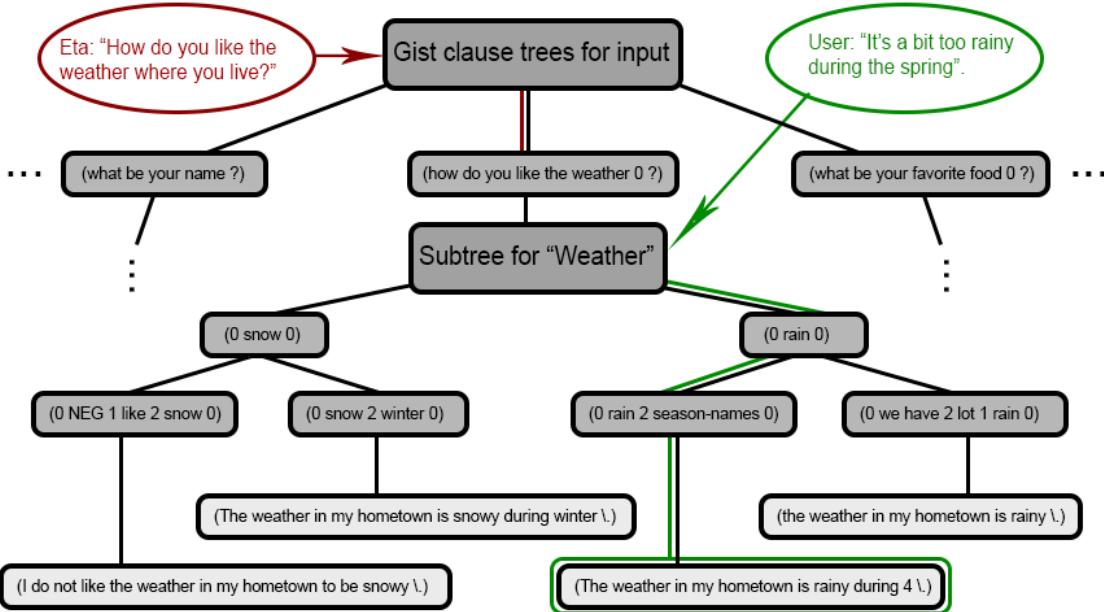


Figure 4.4: An example pattern transduction tree for gist clauses, with a trace for specific example inputs shown.

return ULF formula constituents, which are then composed into larger expressions by the “calling” tree, and returned. At the level of individual words (or certain phrases), a lexicon and lexical routines supply word ULFs. The efficiency and accuracy of the approach lies in the fact that hierarchical pattern matching can quite accurately segment utterances into meaningful parts, often relying on automatically added syntactic and semantic features, so that the need for recursive backtracking rarely arises.

An example of a transduction tree being used for parsing a historical question into ULF is shown and described in Figure 4.5. As in the example mentioned above, the resulting ULF retains much of the surface structure, but uses semantic typing and adds operators to indicate plurality, tense, aspect, and other linguistic phenomena. Additional regularization is done with a limited reference resolution module, which can resolve anaphora and referring expressions such as “it”, “that block”, etc., by detecting and storing discourse entities in context and employing recency and syntactic salience heuristics. The resulting interpretations are added to dialogue context.

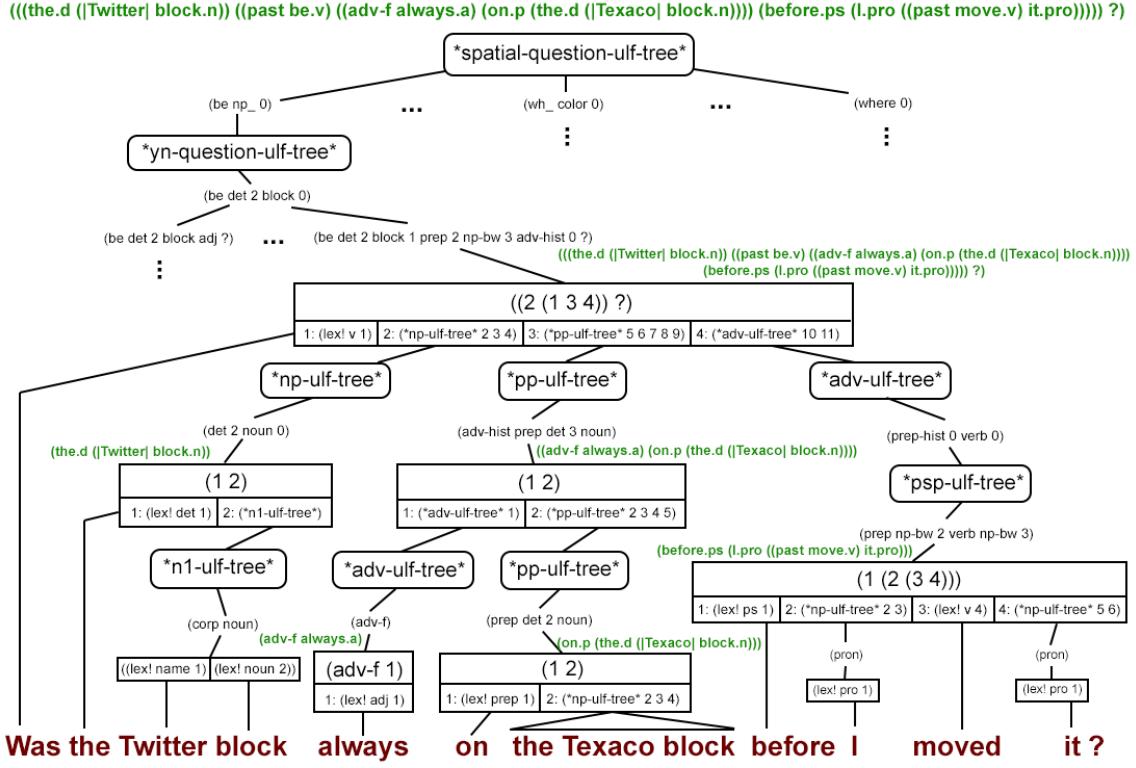


Figure 4.5: An example ULF parse, with the input shown in red, and the resulting ULF (at each composition step) shown in green. The nodes with rectangles represent ULF composition nodes, where the numbers in the upper box correspond to the indices of the lower boxes (if there is no upper box, the constituent ULFs are simply concatenated). All other nodes are patterns to be matched to the corresponding span of input text.

4.2.4 Plan Execution

If the currently pending step in the plan is an expected user action, the system attempts to **match the expectation** to a “currently true” fact in context, unifying any variables in the schema formula. If no fact is matched, the plan will not proceed unless some *waiting threshold* (i.e., elapsed time since the last plan modification) determined by the certainty of the expected episode is exceeded. Certainties are mapped to waiting thresholds according to the function $T(c) = -\alpha * \log(1 - c)$, where α is a global parameter controlling the scale of the resulting times. This choice of function causes $T(0) = 0$ and $\lim_{c \rightarrow 1} T(c) = \infty$, as desired. In the case where $c = 1$, the system will halt indefinitely for the expected input.

In the case of a system action, the system will **execute the action**. If the action is non-primitive, it will select the appropriate subschema (if applicable) and add it

as a subplan to the currently active plan, or otherwise modify the plan directly. If the action is primitive, it can have any of a number of effects depending on the implementation of the execution function for that action: it might modify facts in the context or memory; it might request domain-specific reasoning from a specialist subsystem; it might request a perceptual subsystem to attend to a specific entity; or it might trigger response generation.

In both cases, any plan variables bound during the process of matching an expected episode or executing an action will be substituted in the current plan, as well as an attached copy of the schema used to instantiate the plan (if applicable).

Finally, the goal of a schema may be used to modify the plan in two respects: first, if facts stored in context already satisfy the goal of some subschema in the plan, the system will consider that episode ‘obviated’ and skip over it. Second, if the end of a schema is reached and the goal of the schema still isn’t satisfied, the system can attempt to replan according to some backup strategy (currently, the backup strategies are specified using a pattern transduction tree, and may consist of generating a response, activating a new subschema, or by default simply moving on).

4.2.5 Response Generation

In the simplest setting, the **response generation** module uses the same hierarchical pattern transduction methods that are used for gist-clause interpretation, except a system reaction is selected in the context of the user’s previous gist-clause. As in the case of interpretation, the terminal nodes of transduction trees specify directives, which may include directly outputting some template, or selecting a subschema to activate and react with.

In more complex settings where the system derives deeper ULF semantic interpretations of the user, response generation may require interaction with context and memory (in the case of queries about historical states of the world, for instance) to generate reactions, as well as requiring other syntactic transformations prior to conversion to surface-form utterances. This process is currently handled by a domain-specific module capable of using episodic memory to answer historical questions about objects in a physical “Blocks World” domain, details of which can be found in (Kane et al., 2020). Note that the conversion from ‘final’ ULF representations of responses to surface-form utterances is itself pretty straightforward due to the close correspondence between ULF and natural English; a distinct advantage of ULF over other more opaque semantic representations.

4.2.6 Task Queue

The Eta dialogue manager relaxes the assumption of strictly sequential plan execution and observation that hampered the LISSA system. Plan execution and observation, as well as inferences of various types, may be interleaved in the Eta dialogue manager through the use of a *task queue* that repeatedly cycles between the tasks labelled in green in Figure 4.3, and described in more detail below:

1. **perform-next-step**: handles the currently pending step in the dialogue, performing each of the functions described in Subsections 4.2.4 and 4.2.5 (if applicable).
2. **perceive-world**: collects perceptions about the external world (as symbolic logical representations) from all perceptual subsystems. Perceptions are added to the fact store, as well as added to a queue of new perceptions for further interpretation.
3. **interpret-perceptions**: the system attempts to arrive at a semantic interpretation of new perceptions, using the context of the relevant previous action in the current plan, via the methods outlined in 4.2.3. The interpretations are added to the fact store.
4. **infer-facts**: the system applies an inference process to its fact store to derive new facts in a bottom-up manner. To make this tractable with larger fact stores, the system may use particular strategies to narrow down candidates for inference. Currently, the system only applies inference rules once to “new” perceptions and semantic representations. Transduction trees are also used for inference currently, though in the future this process may involve formal inference rules, or a more data-driven approach.
5. **find-schemas**: the system’s fact store is used to select new candidates for the active schema, and/or to instantiate a candidate as the active schema. The new active schema may either replace the old one completely, or form a subplan. This process is unimplemented in the current version of the system, and remains future work.

4.3 Virtual Human Applications using Eta

The functionality of the Eta dialogue manager is tested in two practical domains, both of which require capabilities beyond those of the LISSA system. The first domain is the virtual avatar *David*, a spatially-aware conversational agent capable of

carrying out various collaborative tasks in a physical “Blocks World” setting. These tasks include question-answering, concept tutoring, and (as an anticipated future goal) concept learning. The second domain is the SOPHIE avatar - a Standardized Online Patient for Healthcare Interaction Education, aimed at allowing medical students to practice having difficult conversations with patients.

4.3.1 Blocks World Avatar (David)

The David virtual agent is intended to hold collaborative dialogues with a user within a physical “Blocks World” setting, depicted in Figure 4.6a. Blocks can be referred to by company names as well as by color (indicated by the tape around the blocks); the scene is captured by two Kinect sensors and reconstructed in the Blender 3D modelling program. The full dialogue pipeline, shown in 4.6b, builds upon the core Eta architecture (diagrammed in 4.3) in a fairly straightforward and standardized way: both the ASR module and the “vision”/state-tracking component of the Blocks World system are connected to Eta as *perceptual subsystems*, while the “spatial component” is connected to Eta as a *specialist subsystem*, capable of task-specific reasoning (e.g., determination of spatial relations between blocks, spatial planning, etc.).

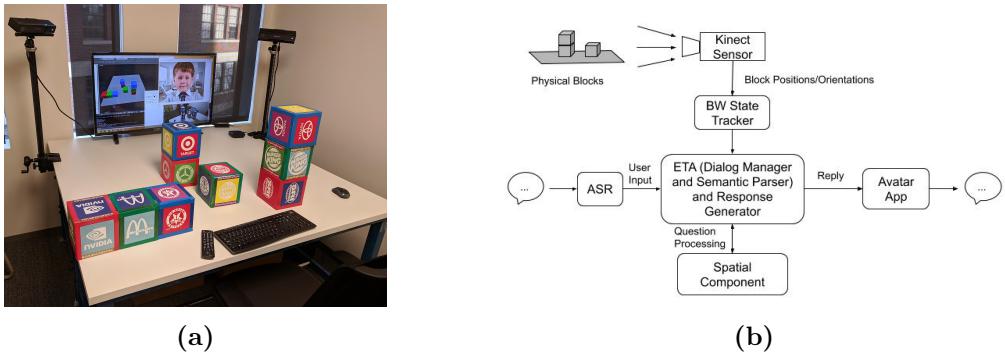


Figure 4.6: The blocks world apparatus setup (a), and dialogue pipeline (b). In (b), the arrows indicate the direction of interaction between the modules.

The initial implementation of the David virtual agent was intended to support open-ended natural language question-answering (QA) about the blocks on the table (Platonov et al., 2020), or about historical states of the blocks or actions by users (Kane et al., 2020). This system is in some sense a spiritual successor to the seminal SHRDLU program (Winograd, 1972), but differs in at least a couple respects: first, the SHRDLU system used a virtual Blocks World, while our system is intended to operate with a physical Blocks World (though it is compatible with a simulated

```

David: Would you like to ask me a spatial question?
User: (moves a block)
User: David, which block did I just move?
David: You moved the Toyota block.
User: Where was the Toyota block before?
David: The Toyota block was between the Mercedes block
      and the Burger King block.
User: Where is it now?
David: The Toyota block is on top of the Texaco block.
User: (moves another block)
User: How many blocks have I moved since the beginning?
David: You moved two blocks.
User: Was the Twitter block always behind the Mercedes block?
David: No.
User: Did I move the Twitter block after I moved the Toyota block?
David: Yes.
User: When did I move the Toyota block?
David: You moved the Toyota block three minutes ago.

```

Figure 4.7: Example of the sort of dialogue that David can engage in (in the QA setting).

world as well); this setting introduces a number of challenges that are not encountered within a simulated setting. Second, our project is intended to answer questions using fully general mathematical models of spatial relations (such as “near” or “behind”) that can be re-used in more realistic domains. Our work also differs from more recent work on collaborative Blocks World dialogues such as (Perera et al., 2018), which focuses on interactively learning concepts through positive and negative examples.

Meeting the requirements imposed by the Blocks World QA task required making many improvements to the underlying dialogue manager. First, as discussed in Section 4.2, being able to accurately answer spatial questions required moving from superficial gist-clause transduction to implementing deep (though domain-specific) compositional semantic parsing using transduction trees. Second, for the system to be able to answer questions about historical states of the world (relevant for, e.g., returning to a previous state or diagnosing a past misstep), it had to be extended with the episodic memory shown in Figure 4.3, and the response generator had to be modified to use a (fairly general) backtracking algorithm over the memory; see (Kane et al., 2020) for more details.

An example of a transcript from the David avatar in the QA task is shown in Figure 4.7. Current work on the David avatar is focused on allowing it to hold collaborative *concept tutoring* dialogues, where the David avatar has in mind some abstract concept (e.g., an arch), and attempts to interactively teach the user the concept through having them create examples with blocks. Furthermore, we’re working

```

SOPHIE: One thing I've noticed is that I haven't been sleeping well.
      Do you know why I keep waking up at night?
User: What happens when you try to sleep?
SOPHIE: Sometimes I wake up in the middle of the night and notice an
      ache in my back and near my chest. I have to take the pain
      medication to fall back to sleep.
User: What pain medication?
SOPHIE: I'm just taking the Lortab for the pain right now.
User: I see.
SOPHIE: It sounds like my pain is the root of my sleeping problems.
      Do you think having stronger pain medication would help?
User: We can certainly try putting you on a stronger medication.
SOPHIE: It would be nice to be able to sleep soundly again. What would
      the side effects of a stronger pain medication be, though?
User: You might face some problems with nausea.
SOPHIE: I think I should try the medicine and see if I have any
      problems with nausea. Could you prescribe me some?
User: Sure, I can do that.
SOPHIE: Thank you very much, I hope that will help with the pain.
      Let's talk about my test results...

```

Figure 4.8: Example of part of a dialogue with the SOPHIE avatar.

on allowing David to *explain* why it gave particular answers in the QA task; this involves decomposing the spatial relations into various low-level *factors* that the system is able to reason with.

4.3.2 Standardized Patient Avatar (SOPHIE)

The SOPHIE (Standardized Online Patient for Healthcare Interaction Education) avatar is a virtual patient recently diagnosed with lung cancer and seeking medical advice, allowing medical students and oncologists to practice having difficult conversations with patients (Ali et al., 2021). In a similar vein as the LISSA study, the SOPHIE system provides post-conversation feedback to the user on several metrics that have been found to be positively correlated with patient prognosis understanding: lecturing vs. non-lecturing, speaking rate, number of questions asked, and sentiment trajectory of the conversation. The dialogue contents were meticulously designed based on guidance from palliative care experts, and transcripts from previous training sessions involving human actors. An example conversation with SOPHIE is shown in Figure 4.8.

On the dialogue manager backend, the system is similar to LISSA and operates mostly on the level of gist-clause interpretation. However, unlike LISSA where dia-

logues were mostly system-initiated, SOPHIE dialogues tend to be mixed-initiative: sometimes the user asks questions with a particular goal in mind, and sometimes SOPHIE needs to ask questions with her particular goal in mind (and ensure that she gets a satisfactory answer). In order to support this, the dialogue manager was extended with limited goal-driven replanning capabilities: dialogue schemas (e.g., asking what one’s prognosis is) may be associated with goals, and if the goal is not satisfied by context at the conclusion of the schema, a transduction tree is used to select some appropriate follow-up behavior (if any; by default the system simply moves on with the current plan). Currently, the only type of goals supported are ones that state that the agent wants to know the answer to a particular utterance.

4.4 Conclusion and Future Work

In this chapter, I discussed the preliminary development of a dialogue manager which is highly modular and portable - generally requiring only modification of hierarchical pattern transduction trees, in a simple English-based templatic language, to create a virtual agent in a new domain - while still capable of deeper understanding and adaptive behavior through the use of a type-coherent, expressive logical form, and flexible dialogue schemas.

Though the core Eta dialogue manager has already shown to be readily applicable to several dialogue domains, much work still needs to be done in order to allow the system to be a fully general, open-domain dialogue system. Some planned future directions of research are as follows:

1. Though the process of producing a semantic interpretation (ULF) from a gist-clause is quite reliable in the Blocks World setting, using a hand-engineered domain-specific compositional parser, it is difficult to extend the parser to other domains. In general, the hand-engineered pattern trees used for gist-clause and ULF interpretation require careful design and can be somewhat rigid. Ultimately, one goal is to use a generic machine learning-based English to ULF parser for interpretation (perhaps with the ability to “override” with reliable domain-specific parsers); to this end, a cache-transition ULF parser is concurrently being worked on by colleagues within my group. The same also applies to response generation.
2. The historical backtracking algorithm used for historical QA in (Kane et al., 2020) was fairly simplistic, relying on an assumption of a linear temporal order of episodes. A more general algorithm would involve traversing a topologically

sorted DAG of episode bounds (i.e., start and end points). A Lisp implementation of this structure - a *Timegraph* - is currently in the works by a collaborator.

3. The ability of the system to react to unexpected inputs is still somewhat limited. To exploit the advantages of *schema-based planning* in handling such events, an algorithm needs to be developed to continuously update a set of *candidate schemas* according to certain *trigger conditions* of the schemas, and to instantiate a new active schema from the set of candidate schemas when appropriate. A related goal, though falling outside the scope of my research, would be automatically learning/extracting a discourse schema library that's broad enough to cover various types of unanticipated behavior.
4. More robust inference rules. Inference currently relies on 'superficial' pattern transduction trees of the same sort used to do gist-clause and ULF interpretation. Support should be added for formal implicational inference rules that derive the conclusion whenever the premise is found to be true (in fact, such inference rules *can* be represented by pattern transduction trees, but requiring a more expressive tree-transduction language than the "flat" pattern language used currently).
5. Similarly, it might be useful to employ statistical natural language inference (NLI) models within the inference pipeline. One task in which NLI models could be useful, for instance, is in deriving "want" and "believe" propositions - ubiquitous as goals and conditions in schemas - from user utterances. For instance, if a user replies to the SOPHIE system with "I think the cancer has spread.", the system should automatically infer "You believe that the cancer has spread.", which can then be used to instantiate a schema with a corresponding precondition. Some separate research that I'm doing with Prof. Aaron White is aimed at training statistical models to analyze such belief/desire inferences.

Finally, and in conclusion, a future research objective is to *combine* the capabilities exhibited by the Blocks World avatar with those exhibited by the SOPHIE and LISSA systems in a single agent with multiple cross-domain competencies. Stated more generally, the ultimate goal of the project is to develop a system which is capable of both broad conversational behavior about casual or domain-specific topics, while simultaneously being "spatially aware" and capable of grounded collaborative discussion about objects in the world. Though many existing dialogue systems are open-domain in the sense that the underlying modules can be easily ported to multiple domains, there are (to my knowledge) no systems which integrate behavior across

multiple domains as just described. Although the goal just described is a highly ambitious one, I believe that the current and future research described in this chapter constitute significant, and attainable, contributions to this goal.

Bibliography

- [1] Daniel Adiwardana et al. *Towards a Human-like Open-Domain Chatbot*. 2020. arXiv: [2001.09977 \[cs.CL\]](https://arxiv.org/abs/2001.09977).
- [2] Mohammad Ali et al. “Novel Computational Linguistic Measures, Dialogue System and the Development of SOPHIE: Standardized Online Patient for Healthcare Interaction Education.” In: *IEEE Transactions on Affective Computing* PP (Jan. 2021), pp. 1–1. doi: [10.1109/TAFFC.2021.3054717](https://doi.org/10.1109/TAFFC.2021.3054717).
- [3] James F. Allen. “Towards a general theory of action and time.” In: *Artificial Intelligence* 23.2 (1984), pp. 123–154. ISSN: 0004-3702. doi: [https://doi.org/10.1016/0004-3702\(84\)90008-0](https://doi.org/10.1016/0004-3702(84)90008-0). URL: <https://www.sciencedirect.com/science/article/pii/0004370284900080>.
- [4] James F. Allen and G. Ferguson. “Human-Machine Collaborative Planning.” In: *Proceedings of the 3rd international NASA Workshop on Planning and Scheduling for Space*. 2002.
- [5] James F. Allen, G. Ferguson, and Amanda Stent. “An architecture for more realistic conversational systems.” In: *IUI '01*. 2001.
- [6] James F. Allen and C.Raymond Perrault. “Analyzing intention in utterances.” In: *Artificial Intelligence* 15.3 (1980), pp. 143–178. ISSN: 0004-3702. doi: [https://doi.org/10.1016/0004-3702\(80\)90042-9](https://doi.org/10.1016/0004-3702(80)90042-9). URL: <https://www.sciencedirect.com/science/article/pii/0004370280900429>.
- [7] John Langshaw Austin. *How to do things with words*. William James Lectures. Oxford University Press, 1962.
- [8] Frederic C. Bartlett and Walter Kintsch. *Remembering: A Study in Experimental and Social Psychology*. 2nd ed. Cambridge University Press, 1995. doi: [10.1017/CBO9780511759185](https://doi.org/10.1017/CBO9780511759185).

- [9] Timothy W. Bickmore, Daniel Schulman, and Candace L. Sidner. “A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology.” In: *Journal of Biomedical Informatics* 44.2 (2011), pp. 183–197. ISSN: 1532-0464. DOI: <https://doi.org/10.1016/j.jbi.2010.12.006>. URL: <https://www.sciencedirect.com/science/article/pii/S1532046411000025>.
- [10] D. Bohus and Alexander I. Rudnicky. “The RavenClaw dialog management framework: Architecture and systems.” In: *Comput. Speech Lang.* 23 (2009), pp. 332–361.
- [11] Julia Campbell et al. “Developing INOTS to support interpersonal skills practice.” In: Apr. 2011, pp. 1 –14. DOI: <10.1109/AERO.2011.5747535>.
- [12] Herbert H. Clark. *Using Language*. ‘Using’ Linguistic Books. Cambridge University Press, 1996. DOI: <10.1017/CBO9780511620539>.
- [13] Herbert H. Clark and Susan E. Brennan. “Grounding in Communication.” In: *Perspectives on Socially Shared Cognition*. Ed. by Lauren Resnick et al. American Psychological Association, 1991, pp. 13–1991.
- [14] Philip R. Cohen and C. Raymond Perrault. “Elements of a plan-based theory of speech acts.” In: *Cognitive Science* 3.3 (1979), pp. 177–212. ISSN: 0364-0213. DOI: [https://doi.org/10.1016/S0364-0213\(79\)80006-3](https://doi.org/10.1016/S0364-0213(79)80006-3). URL: <https://www.sciencedirect.com/science/article/pii/S0364021379800063>.
- [15] G. Ferguson and James F. Allen. “TRIPS: An Integrated Intelligent Problem-Solving Assistant.” In: *AAAI/IAAI*. 1998.
- [16] Richard E. Fikes and Nils J. Nilsson. “Strips: A new approach to the application of theorem proving to problem solving.” In: *Artificial Intelligence* 2.3 (1971), pp. 189–208. ISSN: 0004-3702. DOI: [https://doi.org/10.1016/0004-3702\(71\)90010-5](https://doi.org/10.1016/0004-3702(71)90010-5). URL: <https://www.sciencedirect.com/science/article/pii/0004370271900105>.
- [17] Tim Finin et al. “KQML as an Agent Communication Language.” In: *Proceedings of the Third International Conference on Information and Knowledge Management*. CIKM ’94. Gaithersburg, Maryland, USA: Association for Computing Machinery, 1994, pp. 456–463. ISBN: 0897916743. DOI: <10.1145/191246.191322>. URL: <https://doi.org/10.1145/191246.191322>.
- [18] Lucian Galescu et al. “Cogent: A Generic Dialogue System Shell Based on a Collaborative Problem Solving Model.” In: *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 400–409. DOI: <10.18653/v1/W18-5048>. URL: <https://www.aclweb.org/anthology/W18-5048>.

- [19] B. Grosz and C. Sidner. “Plans for Discourse.” In: 1990.
- [20] Barbara J. Grosz and Sarit Kraus. “Collaborative plans for complex group action.” In: *Artificial Intelligence* 86.2 (1996), pp. 269–357. ISSN: 0004-3702. DOI: [https://doi.org/10.1016/0004-3702\(95\)00103-4](https://doi.org/10.1016/0004-3702(95)00103-4). URL: <https://www.sciencedirect.com/science/article/pii/0004370295001034>.
- [21] Barbara J. Grosz and Candace L. Sidner. “Attention, Intentions, and the Structure of Discourse.” In: *Comput. Linguist.* 12.3 (July 1986), pp. 175–204. ISSN: 0891-2017.
- [22] Vrindavan Harrison et al. *Athena: Constructing Dialogues Dynamically with Discourse Constraints*. 2020. arXiv: [2011.10683 \[cs.CL\]](https://arxiv.org/abs/2011.10683).
- [23] Samuel Humeau et al. *Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring*. 2020. arXiv: [1905.01969 \[cs.CL\]](https://arxiv.org/abs/1905.01969).
- [24] Hans Kamp, Josef Van Genabith, and Uwe Reyle. “Discourse Representation Theory.” In: *Handbook of Philosophical Logic: Volume 15*. Ed. by Dov M. Gabbay and Franz Guenther. Dordrecht: Springer Netherlands, 2011, pp. 125–394. ISBN: 978-94-007-0485-5. DOI: [10.1007/978-94-007-0485-5_3](https://doi.org/10.1007/978-94-007-0485-5_3).
- [25] Benjamin Kane, Georgiy Platonov, and Lenhart K. Schubert. *History-Aware Question Answering in a Blocks World Dialogue System*. 2020. arXiv: [2005.12501 \[cs.AI\]](https://arxiv.org/abs/2005.12501).
- [26] Staffan Larsson et al. “GoDiS - An Accommodating Dialogue System.” In: (Dec. 2002). DOI: [10.3115/1117562.1117564](https://doi.org/10.3115/1117562.1117564).
- [27] Lane Lawley, Gene Louis Kim, and Lenhart Schubert. “Towards Natural Language Story Understanding with Rich Logical Schemas.” In: *Proceedings of the Sixth Workshop on Natural Language and Computer Science*. 2019, pp. 11–22.
- [28] Neal Lesh, Charles Rich, and Candace L. Sidner. “Collaborating with Focused and Unfocused Users under Imperfect Communication.” In: *User Modeling 2001*. Ed. by Mathias Bauer, Piotr J. Gmytrasiewicz, and Julita Vassileva. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 64–73. ISBN: 978-3-540-44566-1.
- [29] Anton Leuski and David Traum. “NPCEditor: Creating Virtual Human Dialogue Using Information Retrieval Techniques.” In: *AI Magazine* 32.2 (2011), pp. 42–56. DOI: [10.1609/aimag.v32i2.2347](https://doi.org/10.1609/aimag.v32i2.2347).
- [30] Karen E. Lochbaum. “A Collaborative Planning Model of Intentional Structure.” In: *Computational Linguistics* 24.4 (1998), pp. 525–572. URL: <https://www.aclweb.org/anthology/J98-4001>.

- [31] Colin Matheson, Massimo Poesio, and David Traum. “Modelling Grounding and Discourse Obligations Using Update Rules.” In: (May 2002).
- [32] M. McTear. “Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit.” In: *ICSLP*. 1998.
- [33] Marvin Minsky. *A Framework for Representing Knowledge*. Tech. rep. Cambridge, MA, USA, 1974.
- [34] Fabrizio Morbini et al. “FLoReS: A Forward Looking, Reward Seeking, Dialogue Manager.” In: (Aug. 2014). DOI: [10.1007/978-1-4614-8280-2_28](https://doi.org/10.1007/978-1-4614-8280-2_28).
- [35] Ian Perera et al. “A situated dialogue system for learning structural concepts in blocks world.” In: *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. 2018, pp. 89–98.
- [36] Ian Perera et al. “Natural Language Dialogue for Building and Learning Models and Structures.” In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI’17. San Francisco, California, USA: AAAI Press, 2017, pp. 5103–5104.
- [37] Georgiy Platonov et al. “A Spoken Dialogue System for Spatial Question Answering in a Physical Blocks World.” In: *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 1st virtual meeting: Association for Computational Linguistics, July 2020, pp. 128–131. URL: <https://www.aclweb.org/anthology/2020.sigdial-1.16>.
- [38] Massimo Poesio and David Traum. “Towards an Axiomatization of Dialogue Acts.” In: (May 1998).
- [39] Massimo Poesio and David R. Traum. “Conversational Actions and Discourse Situations.” In: *Computational Intelligence* 13.3 (1997), pp. 309–347. DOI: <https://doi.org/10.1111/0824-7935.00042>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/0824-7935.00042>.
- [40] Martha E. Pollack. “Plans As Complex Mental Attitudes.” In: *Intentions in Communication*. MIT Press, 1990, pp. 77–103.
- [41] S. G. Pulman et al. “‘How Was Your Day?’” In: *Proceedings of the 2010 Workshop on Companionable Dialogue Systems*. CDS ’10. Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 37–42. ISBN: 9781932432817.
- [42] S. Z. Razavi et al. “Dialogue Design and Management for Multi-Session Casual Conversation with Older Adults.” In: *ArXiv* abs/1901.06620 (2019).
- [43] S. Z. Razavi et al. “Managing Casual Spoken Dialogue Using Flexible Schemas , Pattern Transduction Trees , and Gist Clauses.” In: 2017.

- [44] Seyedeh Zahra Razavi et al. “The LISSA Virtual Human and ASD Teens: An Overview of Initial Experiments.” In: *Intelligent Virtual Agents*. Ed. by David Traum et al. Cham: Springer International Publishing, 2016, pp. 460–463. ISBN: 978-3-319-47665-0.
- [45] Charles Rich, Candace Sidner, and Neal Lesh. “COLLAGEN – Applying Collaborative Discourse Theory to Human-Computer Interaction.” In: *AI Magazine* 22 (Dec. 2001), pp. 15–26.
- [46] Albert Rizzo et al. “SimCoach: An intelligent virtual human system for providing healthcare information and support.” In: *Studies in health technology and informatics* 163 (Nov. 2011), pp. 503–9. DOI: [10.1515/IJDHD.2011.046](https://doi.org/10.1515/IJDHD.2011.046).
- [47] Stephen Roller et al. *Recipes for building an open-domain chatbot*. 2020. arXiv: [2004.13637 \[cs.CL\]](https://arxiv.org/abs/2004.13637).
- [48] R. Schank and M. Burstein. “Artificial Intelligence: Modeling Memory for Language Understanding.” In: *Handbook of Discourse Analysis. Volume 1: Disciplines of Discourse*. Ed. by T. A. van Dijk. London: Appleton Century Crofts, 1985, pp. 145–166.
- [49] R.C. Schank and R.P. Abelson. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. The Artificial Intelligence Series. Lawrence Erlbaum Associates, 1977. ISBN: 9780470990339. URL: <https://books.google.com/books?id=YZ99AAAAMAAJ>.
- [50] Lenhart K Schubert and Chung Hee Hwang. “Episodic Logic meets Little Red Riding Hood: A comprehensive, natural representation for language understanding.” In: *Natural language processing and knowledge representation: Language for Knowledge and Knowledge for Language* (2000), pp. 111–174.
- [51] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969. DOI: [10.1017/CBO9781139173438](https://doi.org/10.1017/CBO9781139173438).
- [52] D. Traum and James F. Allen. “Discourse Obligations in Dialogue Processing.” In: *ACL*. 1994.
- [53] David R. Traum and Staffan Larsson. “The Information State Approach to Dialogue Management.” In: *Current and New Directions in Discourse and Dialogue*. Ed. by Jan van Kuppevelt and Ronnie W. Smith. Dordrecht: Springer Netherlands, 2003, pp. 325–353. ISBN: 978-94-010-0019-2. DOI: [10.1007/978-94-010-0019-2_15](https://doi.org/10.1007/978-94-010-0019-2_15). URL: https://doi.org/10.1007/978-94-010-0019-2_15.
- [54] Roy M. Turner. *Adaptive Reasoning for Real-World Problems: A Schema-Based Approach*. Hillsdale, NJ: Lawrence Erlbaum, 1994.

- [55] Andrew Valenti et al. “A Multi-level Framework for Understanding Spoken Dialogue Using Topic Detection.” In: *Proceedings of the 24th Workshop on the Semantics and Pragmatics of Dialogue - Full Papers*. Virtually at Brandeis, Waltham, New Jersey: SEMDIAL, July 2020. URL: http://seminal.org/anthology/Z20-Valenti_semidial_0023.pdf.
- [56] Teun Adrianus Van Dijk and Walter Kintsch. *Strategies of discourse comprehension*. Academic press New York, 1983.
- [57] Joseph Weizenbaum. “ELIZA—a computer program for the study of natural language communication between man and machine.” In: *Communications of the ACM* 9.1 (1966), pp. 36–45.
- [58] Chris Wilson and Roy M. Turner. “Context and the Virtual Human.” In: *Modeling and Using Context - 10th International and Interdisciplinary Conference, CONTEXT 2017, Paris, France, June 20-23, 2017, Proceedings*. Ed. by Patrick Brézillon, Roy M. Turner, and Carlo Penco. Vol. 10257. Lecture Notes in Computer Science. Springer, 2017, pp. 227–239. DOI: [10.1007/978-3-319-57837-8_18](https://doi.org/10.1007/978-3-319-57837-8_18). URL: https://doi.org/10.1007/978-3-319-57837-8_18.
- [59] Terry Winograd. “Understanding natural language.” In: *Cognitive psychology* 3.1 (1972), pp. 1–191.
- [60] Wei Xu and Alexander I. Rudnicky. “Task-based dialog management using an agenda.” In: *ANLP-NAACL 2000 Workshop: Conversational Systems*. 2000. URL: <https://www.aclweb.org/anthology/W00-0309>.