

Computer Science Thesis Proposal

Schema-Guided Dialogue Management for Virtual Conversational Agents

Benjamin Kane

Thesis Committee

Dr. Lenhart Schubert Dr. Aaron Steven White Dr. Ehsan Hoque Dr. Scott Grimm

April 2022

University of Rochester Department of Computer Science

Abstract

The development of "intelligent" dialogue assistants and conversational agents for general-purpose use – that is, able to be deployed to various domains with minimal engineering – is a long-standing goal of the AI community. Of particular necessity is an underlying *dialogue manager* capable of seamlessly coordinating several interacting elements: the interpretations of utterances that the agent hears, the agent's own desires and beliefs, and other external events that happen in the world. Furthermore, the agent must perform this coordination in a manner that is adaptable to unexpected events or inputs. The matter of how to mediate these objectives within a scalable dialogue system remains an open problem.

I propose a dialogue management framework that relies on *dialogue schemas* – structured representations of *prototypical* sequences of events in a dialogue, expressed in a type-coherent logical form closely resembling surface English – as a medium for adaptive planning. This dialogue manager combines multiple interdependent components including a dialogue context and episodic memory, a two-stage pipeline for semantic interpretation, a schema-driven planner, and modular subsystems for perception or specialized reasoning.

To evaluate this approach, I discuss two virtual conversational avatars that have been developed using this dialogue manager – a collaborative agent in a physical "blocks world" domain, and a virtual standardized patient that allows doctors to practice patient communication skills – and present initial experimental results from each domain. I describe several related research goals that I aim to pursue in future work, and an approximate plan for achieving these goals.

Contents

1	Introduction					
2	Bac	Background and Related Work				
	2.1	Plan and Schema-Based Theories of Dialogue	6			
		2.1.1 Hierarchical Plans	7			
		2.1.2 Discourse Schemas	14			
	2.2	General-Purpose Dialogue Systems	20			
		2.2.1 COLLAGEN Dialogue Manager	21			
		2.2.2 TRIPS Dialogue System	24			
		2.2.3 Information State Architectures	29			
		2.2.4 Deep Learning-Based and Hybrid Architectures	33			
3	Eta	Dialogue Manager	37			
	3.1	Dialogue Schema Representation	38			
	3.2	System Architecture	41			
	3.3	Dialogue Tasks	42			
	3.4	Perceptual and Specialist Subsystems	43			
	3.5	Facts Store	44			
	3.6	Semantic Interpretation	44			
	3.7	Hierarchical Planner	47			
	3.8	Response Generation	48			
4	App	lications of Eta and Results	49			
	4.1	Blocks World Avatar (David)	49			
		4.1.1 Domain and Experimental Setup	49			
		4.1.2 Question-Answering Task	50			
		4.1.3 Concept Tutoring	54			
	4.2	Discussion	56			
	4.3	Standardized Patient Avatar (SOPHIE)	57			
		4.3.1 Domain and Experimental Setup	57			

		4.3.2	Dialogue Design	58
		4.3.3	Pilot User Experiment	59
		4.3.4	Dialogue Evaluation	60
	4.4	Discus	sion	64
5	Res	earch]	Plan	67
	5.1	Resear	ch Goals	67
		5.1.1	Overarching Goal: Creating a Unified, General-Purpose Con-	
			versational Agent	67
		5.1.2	Exploring Less Brittle Methods for Interpretation & Generation	68
		5.1.3	Leveraging Bottom-Up Inference From Stored Facts	71
		5.1.4	Utilizing Intensional Models for Schema Selection	72
		5.1.5	Tense-trees and Timegraphs for Temporal Reasoning	74
		5.1.6	Dynamic Schema Matching for Handling Unexpected Events .	78
	5.2	Resear	ch Timeline	79
		5.2.1	Fall 2022:	79
		5.2.2	Spring 2023:	80
		5.2.3	Fall 2023:	80
		5.2.4	Spring 2024 and Beyond:	81
6	Con	clusio	1	82

6 Conclusion

Chapter 1 Introduction

In some sense, creating a truly general-purpose, human-like conversational agent 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 or resources (e.g., a planner, ontology, or knowledge base) to coordinate both dialogue and non-dialogue behavior.
- 4. Deciding what action or utterance the system should perform next, and when it should perform it.

Traditionally, the dialogue management techniques of general-purpose conversational systems have been deeply rooted in theories of human dialogue; under this view, human dialogue unfurls as participants in a conversation collaboratively enact *plans*, i.e., sequences of actions that achieve particular goals. Indeed, plans seem to take on first-order significance in human behavior in general: 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. The role of an automated dialogue manager within this paradigm is to reason about *speech acts* (Searle, 1969) – dialogue actions within a plan that produce particular effects (particularly, on the beliefs, desires, and intentions of the interlocutor) when certain preconditions obtain.

While the traditional plan-based view of dialogue provides a powerful framework for dialogue management, it encounters numerous limitations in handling the full scope of human dialogue. Successful execution of dialogue plans typically requires checking beliefs, desires, and intentions of agents against speech act conditions, yet these propositional attitudes are not often directly accessible in dialogue, but rather are revealed through subtle inferences triggered by natural language utterances. Even more critically, people in natural conversation 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. These issues often rendered early conversational systems brittle and topically narrow.

Modern statistical language modelling techniques, in which deep neural networks are trained to generate text by analogy to billions of examples of human dialogues within its training data, provide a partial remedy to the issues that faced classical systems – at the expense of washing out the strengths of classical systems, such as the ability to handle dialogue as deliberate behavior towards some end, rather than mere reproduction of surface forms. I argue that in order to achieve the "best of both worlds", an approach is needed that maintains some sort of explicit semantic representation for dialogue, while allowing for adaptive behavior – and possible integration of statistical techniques – at the representational level.

In this proposal, I describe a dialogue system that uses *dialogue schemas*, together with a multi-stage interpretation and inference process, for adaptive conversational planning. Dialogue schemas are semantic representations of "prototypical" patterns of dialogue events that can be constructed and used to flexibly guide action, in a manner that can be driven by either goal-driven planning or "matching of expectations". As such, schemas provide a useful generalization from plans (Turner, 1994) as well as from other related concepts such as event scripts (Schank and Abelson,

1977).

In Chapter 2, I present background on planning and schemas as employed in dialogue theory, as well as discuss several related dialogue management frameworks for designing virtual conversational agents. In Chapter 3, I present the schema-based dialogue manager around which my proposal is based, beginning with discussing the dialogue schema representation, followed by the architecture and operations of the system. In Chapter 4, I introduce the applied conversational agents that provide the main experimental testbeds for our system – a standardized virtual patient (SOPHIE) for helping doctors practice difficult conversations with patients, and a collaborative conversational agent (David) in a physical "blocks world" domain – and present some initial results for each. Finally, in Chapter 5, I detail my primary research goals and provide an approximate timeline for my future research.

Chapter 2 Background and Related Work

2.1 Plan and Schema-Based Theories of Dialogue

Before there were human-computer dialogue systems, there were attempts to explain and analyze human-human dialogue. Unsurprisingly, the former systems often had their roots in the computational 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, I first provide an overview of some of the foundational theory underlying plan-based and schema-based dialogue research.

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. These models, however, underwent a major conceptual shift in the 80s and 90s: while plans were initially based on classical STRIPS-like operators (Fikes and Nilsson, 1971) for speech acts, they would later be used to represent private internal cognitive attitudes of agents, and eventually to represent collaborative plans that agents jointly enact subject to a common ground.

An alternative to the plan-based theory of dialogue is characterizing dialogue as relying on more reflexive or analogical reasoning based on expected 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.

2.1.1 Hierarchical Plans

STRIPS-like Operators for Speech Acts

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 (sets of ground logical propositions) – these operators can specify preconditions that must to be true in order for the operator to be applied and effects that modify the state of the world after application of the operator. A STRIPS plan consists of a chain of operators that, when applied to some initial state, transform the state into a desired goal state. Typically, this basic formalism is extended with *hierarchical planning*, i.e., the ability to decompose operators into several sub-actions or sub-goals.

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 that operate on individual cognitive states of dialogue participants, such as beliefs and intentions.

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 (i_1) S intends for H to believe that, by uttering T, S intends for them to do A; (i_2) S intends for H to recognize their intention i_1 , and (i_3) S intends for H to be convinced to do A by means of recognition of i_1 .

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 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.

An example of an operator for an **informref** speech act – informing a hearer of the value of some definite description, such as in a 'wh-question' – is shown in Figure 2.1. The preconditions and effects of these operators contain free variables which get 'filled in' as the plan is constructed¹. In order to enable complete planning, the perlocutionary acts are also modelled as operators; in the example of **informref**, a corresponding **convinceref** operator must be created to reach the state where the hearer believes the proposition.

```
INFORMREF(S, H, \lambda x Dx):

CANDO.PR: \exists y \ S \ BEL \ (ixDx) = y

WANT.PR: S BEL S WANT informref-instance

EFFECT: \exists y \ H \ BEL \ S \ BEL \ (ixDx) = y
```

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

Further details of this theory are developed by Allen and Perrault (1980), who provide a tractable algorithm for collaboratively constructing and executing dialogue

¹"ixDx" represents the definite description "the x which is D", where D is a predicate with one free variable.

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 allowing hierarchical planning. 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 $\pi = \langle \Delta_{alt}, \Delta_{exp}, r \rangle$, 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.

Plan inference rules consist of 'if-then' rules that are used by an agent to form beliefs about the interlocutor's intents. For example, if S BEL A WANT P and P is a precondition of some ACT, then S will infer that S BEL A WANT ACT. Similar rules exist for reasoning about action bodies and effects. *Plan construction rules* are used by an agent to construct their own plan (as opposed to inferring beliefs of the interlocutor) and are essentially the reciprocals of plan inference rules. For example, if S WANT ACT and P is a precondition of ACT, then S will infer that S WANT P.

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. Multiple heuristics are used to rate partial plans including decreasing a rating if the partial plan contains an action whose preconditions are currently false, and increasing the rating of a partial plan if its alternative contains propositions that 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 that is executed by the agent.

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, but many human dialogues involve an agent correcting another agent on an invalid plan. These models also assumed a certain role asymmetry in dialogue: the speaker is assumed to have 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 "masterslave 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 agents holding beliefs about plan operators that can be correct or incorrect.

Pollack argues that for an agent A to have a plan to do act β is really to have a set of conditions such that: A believes that A can execute each sub-act α of β ; A believes that executing these acts will entail A's performance of β ; A intends to do each sub-act of β , and so on. Moreover, A must believe and intend that by doing some sub-act α , they directly or indirectly contribute to doing β – where "contribute" may be cached out in terms of either generation (e.g., by uttering "It is cold in here" under certain conditions, A generates the act of requesting a window to be closed) or enablement.

With these concepts in hand, Pollack (1990) presents a formal theory of *SimplePlans*: to assert that an agent has a plan is to assert several predications about the beliefs and intentions of that agent corresponding to the conditions described above². The primary insight of this formalism is that it allows us to talk precisely about what it means for an agent A to have an invalid plan: A has the set of beliefs

 $^{^{2}}$ In her presentation of SimplePlans, Pollack relied on Allen's temporal logic (Allen, 1984) to formally represent these conditions.

and intentions described previously, 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 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 corrects another, or suggests some sort of plan modification.

Plans as Joint Activity 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. An alternative is to view dialogue as a joint activity. The SharedPlan formalism – developed and refined over the course of several publications (Grosz and Sidner, 1990; Grosz and Kraus, 1996; Lochbaum, 1998) – presents an alternative approach where dialogue participants collaboratively build and augment a joint plan by contributing information about their beliefs, wants, and individual plans through utterances.

This formalism is an extension of (Pollack, 1990) in the sense that agents can have Individual Plans (IPs) – i.e., SimplePlans as defined previously – as well as Shared Plans (SPs). Much like the case of SimplePlans, for a group GR to have an SP is really to have a set of conditions on the group and its agents³:

- 0. GR is committed to performing β (and has a recipe for doing so)
- 1. For each single-agent act α_i in the recipe, there is an agent $G_{\alpha_i} \in GR$ such that:
 - (a) G_{α_i} intends to, and believes it can, perform α_i , and has an IP for α_i
 - (b) The group GR mutually believe (2a)
 - (c) The group GR is committed to G_{α_i} 's success
- 2. For each multi-agent act α_i in the recipe, there is a subgroup of agents $GR_{\alpha_i} \subseteq GR$ such that:
 - (a) GR_{α_i} mutually believe that they can perform α_i , and have an SP for α_i
 - (b) The group GR mutually believe (3a)
 - (c) The group GR is committed to GR_{α_i} 's success

³As with SimplePlans, these conditions are formally represented using Allen temporal logic.



Figure 2.2: The three components in the tripartite model of discourse structure.

Agents can also have *partial* IPs or SPs in which only a subset of the necessary conditions hold. The process of collaborative reasoning with SharedPlans involves *augmenting* a partial plan to form a full plan, with individual utterances contributing to the augmentation process through formation of mutual beliefs.

At the core of the SharedPlan approach is a tripartite model of discourse structure first introduced by Grosz and Sidner (1986), consisting of a hierarchical structure of utterances in the discourse, a "dominance hierarchy" of discourse-level intentions (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 intention for that discourse segment). These three components are illustrated in Figure 2.2.

The components in this model directly correspond to relations among Shared-Plans for dialogue acts – specifically, if some DSP dominates another DSP, then there's a subsidiary relationship between the SharedPlans corresponding to these discourse segments. Similarly, partial and complete SharedPlans correspond to partial and incomplete discourse segments. 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 (1998) 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). However, she also notes some possible limitations of this approach for creating practical collaborative systems – a topic that I revisit in Section 2.2.1.

Common Ground 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.

Moreover, many aspects of ordinary conversation (including the collaborative kind) are difficult or impossible to analyze using these 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 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.

An extension to classical speech act theories of dialogue – the *information state* formalism – posits a structured formal representation of the "conversational score" including both information that's been grounded by participants (i.e., added to common ground) and ungrounded "contributions" or discourse units (Poesio and Traum, 1998). This information state may additionally include discourse obligations (Traum and Allen, 1994), turn-taking status, discourse structure, and shared or individual cognitive states and plans (subsuming the previous theories). Dialogue acts are then viewed as *updates* to the information state, with associated conditions and effects. The original information state formalism described by Poesio and Traum (1998) was based on Compositional DRT (Muskens, 1995) with information states being formally represented as Discourse Representation Structures (DRSs), but computational implementations based on this formalism would often use simpler logical representations with records, lists, sets, or other typed structures in place of DRSs. I return to the

information state approach in Section 2.2.3, where I show a computational example and describe some systems that use this formalism.

2.1.2 Discourse Schemas

Though plan-based formalisms, 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. Since possible changes in contextual knowledge are endogenous to the dialogue planning process in these theories (i.e., as conditions/effects within rules or operators), it imposes a trade-off between an agent exhibiting the same behavior regardless of context on one extreme, and on the other extreme resulting in an intractable combinatorial explosion of rules/operators with each instance conditioned on a different context (Wilson and Turner, 2017).

An alternative formalism for dialogue relies on the concept of "schemas". Schemas are similar to hierarchical plans, but represent 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 3, I discuss my group's specific schema language.

Schema Background

The concept of a "schema" has 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 paper⁴, 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.

Discourse Analysis with Schemas

One of the earliest attempts to use schemas in discourse analysis appeared in the formalism proposed by 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 dis-

 $^{{}^{4}}$ See (Lawley et al., 2019) for more details.



Figure 2.3: A diagram of the discourse theory presented by Van Dijk and Kintsch (1983).

course. The model describes higher levels of the macrostructure as being derived from sequences of propositions in lower levels through different forms of abstraction (e.g., deleting non-relevant propositions, or generalizing several propositions into a single one entailed by each).

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 2.3. 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 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.

Context-Mediated Behavior with Schemas

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.

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 domainindependent 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 2.4. 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.



Figure 2.4: 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 2.5: The schema-based reasoning process diagrammed by Turner (1994), with annotated steps.

The schema-based reasoning process allows for dynamic modification of a dialogue or plan as external events are observed. This process is diagrammed in Figure 2.5. 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 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.

2.2 General-Purpose Dialogue Systems

In this section, I turn my attention towards reviewing seminal efforts in creating general-purpose dialogue systems for development of mixed-initiative virtual conversational agents. 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 or mixed-initiative dialogues. On the other hand, the 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 goaldirected behavior and thus are generally insufficient for many practical applications. Nevertheless, I do discuss a state-of-the-art deep learning-based dialogue system in Section 2.2.4 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.

Though many of the systems presented in this section are applied or tested in specific domains, they are 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 the system with domain-specific rules or knowledge – the automation of which is an as-of-yet unsolved task in AI).

2.2.1 COLLAGEN Dialogue Manager

One of the first influential general-purpose dialogue architectures was the COLLA-GEN 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, "Why did you/we (not do) ...?", "How do I/we/you do ...?", etc. One early application of COLLAGEN, for instance, was a virtual agent acting as an assistant to help a user set up and program a video cassette recorder (VCR). The user could ask questions such as "What next?", and the system might respond with "Now press Play on your camcorder...".

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 2.6.

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



Figure 2.6: 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 2.7: Example of a COLLAGEN recipe for recording a program (in a Java-based syntax).

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 2.7.

One further innovation of the COLLAGEN system was its plan recognition module used for discourse interpretation, 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 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. The overall system architecture combining these components is shown in Figure 2.8.



Figure 2.8: A diagram of the architecture of the COLLAGEN dialogue system.

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 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. Moreover, 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.

Notable Extensions of the COLLAGEN Approach

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 including (but not limited to) knowledge about different theories of behavioral medicine, how the associated interventions relate to specific health behaviors, knowledge about particular users, etc. 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 handdesigned 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.

RavenClaw Dialogue Manager Though developed independently to the COL-LAGEN system, RavenClaw (Bohus and Rudnicky, 2009) employs a very similar dialogue model based on 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 mixedinitiative 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.

2.2.2 TRIPS Dialogue System

The original TRIPS system was designed contemporaneously with the COLLAGEN system. 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.



Figure 2.9: The overall architecture of the TRIPS dialogue manager.

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

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 2.9 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 2.1, along with examples of each from a hypothetical TRIPS dialogue.

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

Table 2.1: 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.

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 2.1. 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.
- 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 at hand; being an interface between the generic problem-solving actions used by the Behavioral Agent (e.g., "create a solution") and the actual task-specific performance of those actions; and assisting the interpretation manager in recognizing user intents when the user performs an 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 problemsolving 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.

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 CO-GENT 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* (ACIs). 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.

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 CO-GENT 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).

2.2.3 Information State Architectures

One more general class of dialogue managers to be discussed are those based on the information state formalism described in Section 2.1. The formalism is realized in computational systems through the inclusion of the following elements:

- 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.
- 2. A set of **dialogue moves** that update the information state, and rules for recognizing/realizing performance of these moves.
- 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.

A simple example of this approach is shown in Figure 2.10, 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), 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 the propositional content in common ground resolves the question under discussion, the QUD stack is popped in step (4) using **downdateQUD**.



Figure 2.10: 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.

Although this example is simplistic, the representation is quite flexible and can be extended with various additional constructs, and has been used to create both closed-task dialogue systems (Larsson et al., 2002; Matheson et al., 2002) as well as general-purpose dialogue managers. I discuss some of the latter below.

Notable Extensions of the Information State Approach

'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 several novel contributions: the use of "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.).

After collecting input from three sources – affect annotation, automatic speech recognition, and emotional speech recognition – the system segments the inputs and tags them with dialogue acts (including detecting turn-taking information), and then sends them to a natural language understanding module. Part-of-speech tagging and named entity recognition are performed using an HMM and hand-engineered patterns, and the chunks are then combined compositionally using a chart parser to form parse trees. The dialogue manager maintains an information state containing all of the objects detected in this pipeline, as well as conversation goals – if a new entity is recognized and introduced by the user input, the DM updates the information state with a goal to talk about it with a corresponding importance rating (calculated using heuristics such as recency).

The system uses a variety of strategies to generate empathetic responses, including sentiment analysis and consultation with a knowledge base to produce an emotional model. These components are used to produce an "affective strategy" (i.e., a template for a longer emotive response) that allows the system to generate responses that are appropriate given the user's expressed sentiments and object mentions.

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.

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 ($D_{KL}(P(F) || P(W))$), 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.

FLoReS Dialogue Manager The FLoReS (Forward Looking, Reward Seeking) dialogue manager was created to overcome one of the main limitations of other systems developed with the information state formalism – namely, their dependence on strong single-initiative dialogue. These architectures were limited in more general multi-initiative dialogue, where the user may question the system, change the topic, or otherwise act in unexpected ways. The FLoReS system addresses this issue (while still allowing for the ease of customization that characterized precursors) by combining several methods of dialogue reasoning as well as a reward-seeking algorithm for determining dialogue policy (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 by extending the basic information state approach with *local subdialogue networks* (i.e., operators) for specific conversation topics – represented as a tree of system and/or user actions and resulting states. At any point, operators can be classified as *currently active*, *paused*, or *inactive*; as events are received, the dialogue manager updates the information state and then decides which operator to use to deal with the received event (it may continue with the currently active operator, or switch to a promising paused/inactive one).

This decision is handled dynamically at runtime according to a probabilistic reward-seeking algorithm. The expected reward for choosing a given operator while at a particular information state is determined by constructing a DAG of possible future dialogues that can happen if that operator is selected, with nodes corresponding to information states and edges to operators. Each node is assigned a probability according to the expected reward associated with that operator, calculated using an algorithm for reward propagation (where information states are assigned rewards based on the priority of the goals within them).

Discussion One strength of the systems built using the information state approach is the ease in which dialogue systems can be designed for various counselling and conversational practice domains, with agent behavior readily customizable by nonexperts. Furthermore, aspects of this approach are amenable to integration 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.

2.2.4 Deep Learning-Based and Hybrid Architectures

Since the advent of deep learning, many recent dialogue systems try to improve the system's ability to generate robust and realistic responses by leveraging statistical models trained on massive dialogue corpora. Some systems attempt to create topically broad conversational systems based on end-to-end Transformer architectures. Although such systems fall mostly outside of the scope of my proposal due to their lack of dialogue management capabilities, I discuss some state-of-the-art systems that were able to achieve robust behavior through specialized training procedures – while these systems are not yet useful for complex mixed-initiative domains, it's possible that future work could use such systems for language generation in addition to a goal-based planner. I also discuss a few hybrid systems that combine planning capabilities with statistical sub-modules.

End-to-End Transformer-Based Architectures

Two particular systems – Google's Meena system (Adiwardana et al., 2020) and FAIR's Blenderbot system (Roller et al., 2020) – are able to achieve robust, topically broad conversational abilities through a combination of specialized Transformer architectures and particular conversation-oriented training datasets. I will focus on the latter system, as it slightly edges out the former in various benchmark metrics. An example of a dialogue with the Blenderbot system is shown in Figure 2.11.

Roller et al. (2020) compare three model architectures for the dialogue system: a **Generator** model that implements a standard Seq2Seq Transformer architecture; an attention-based **Retriever** model that uses the dialogue history as context to select the next dialogue utterance from a candidate set (typically, all possible training set responses)⁵; and a **Retrieve and Refine** model that combines the previous two models – a retrieval model is first used to select a candidate response, which is then paraphrased by the generator model.

 $^{{}^{5}}$ In a sense, this model is a "higher firepower" version of the statistical retrieval system used by the NPC Editor response planner (Leuski and Traum, 2011), with the trade-off of requiring vastly larger amounts of data to use effectively.



Figure 2.11: A "meta-cherrypicked" example dialogue with FAIR's Blenderbot system.

Each model is pre-trained using 1.5B training examples derived from online message board discussions. The authors fine-tune with several datasets: the *ConvAI2* dataset where crowd-workers have casual "getting to know you" conversations; the *Empathetic Dialogues* dataset where one crowd-worker describes a personal situation and the other responds empathetically; the Wizard of Wikipedia dataset where crowd-workers hold conversations while utilizing wikipedia facts from a randomly assigned topic; and the *Blended Skill Talk*, where a "guided" participant is able to select utterances from chatbots trained on the previous three datasets.

Using a crowdsourced transcript comparison protocol for evaluation, the authors find that using the Blended Skill Talk dataset, along with minimum length beamsearch decoding with repetition penalties, produces the highest-scoring dialogues.

Despite the monumental realism of the system's responses in the "cherry-picked" sample transcripts, the authors note several limitations of their system including a tendency to contradict itself or forget a fact previously mentioned in the conversation, a tendency to "hallucinate" false knowledge, fairly short conversation lengths, and a lack of deep semantic understanding or grounding. 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.

Hybrid Architectures

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 incorporating statistical submodules, e.g., for topic-detection in response generation or interpretation.

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 natural language understanding pipeline 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 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 dialogue 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 response generators 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.

Spoken Dialogue with Statistical Topic Detection The dialogue system described in (Valenti et al., 2020) extends a plan-based dialogue manager with a multilevel approach to interpretation. First, upon receiving a list of scored hypotheses (i.e., transcriptions of the user's spoken utterance) from the speech recognition component, the system uses a *topic selection* model 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. This allows the system to more effectively avoid errors in the case of atypical or out-of-domain utterances.

After a 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 speech recognition system that causes it to change prior probabilities on words, making certain hypotheses more likely based on recognized context.

Schema Attention Model Mehri and Eskénazi (2021) argue that the zero-shot generalizability of end-to-end systems such as those described above (that is, their ability to generalize to particular conversational domains) is hampered by the fact that these models are forced to memorize domain-specific dialogue policies. To this end, they propose a Schema Attention Model where the task-specific dialogue policy is explicitly provided to the language model in the form of a schema.

The schema representation used in this paradigm consists of a DAG where nodes correspond to system actions and user utterances, and edges to possible transitions between these. The DAG is deterministic in the sense that each user utterance corresponds to exactly one system action. Thus, the role of the schema attention model is to compute an *alignment* of the dialogue context to a schema node (both encoded using contextualized embeddings) using word-level attention. The resulting alignment scores are used to select and execute the schema node with highest probability.

This schema representation differs significantly from the notion of schemas in my proposal – which instead represent *prototypical* sequences of events (and other expected conditions) and may not necessarily be deterministic. Nevertheless, this work is relevant to my proposal in that it shows that language models can be effectively used as a tool for matching/aligning a dialogue context with plan-like dialogue policies – an idea discussed further in the context of my research plan in Chapter 5.

Chapter 3 Eta Dialogue Manager

In this chapter, I describe the general-purpose schema-based dialogue manager - Eta – that forms the core of my past and proposed research.

The Eta dialogue manager was initially borne out of the LISSA (Live Interactive Social Skill Assistance) dialogue system (Razavi et al., 2016; Razavi et al., 2019), 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.) – this system was used in studies on teenagers with Autistic Spectrum Disorder (ASD) (Razavi et al., 2017), as well as elderly patients at risk of isolation (Razavi et al., 2019). While these studies yielded promising results, the rigid dialogue planner and shallow interpretive pipeline of the LISSA system limited it to superficial system-initiated dialogues. In order to allow for more complex mixed-initiative dialogues and collaborative planning domains, the Eta dialogue manager extends LISSA with numerous capabilities including a dialogue context and episodic memory, deeper semantic parsing, and task cycling between multiple independent behaviors. These details are discussed further in the following sections.

The Eta dialogue manager is also conceptually similar to many of the systems discussed in Chapter 2. It is similar to SharedPlan and TRIPS-based systems in that it maintains a hierarchical dialogue plan consisting of anticipated actions by the user and system, to be jointly instantiated as the conversation proceeds. However, the plan itself is subject to "on the fly" modification as unexpected events are observed or a new schema is matched – in this sense it is similar to the schema-based planning systems described by Turner (1994), but whereas the schema representations used in those systems relied on slots with agglutinated propositional values (e.g. "obstacle-avoidance-active") or simple extensional logics, Eta uses a logical form rich enough to match the expressivity of natural language. Eta also resembles the *information*

state approach in that it maintains a dialogue state consisting of common ground/context, system memory, dialogue history, etc. which is subject to modification by observations and agent actions.

3.1 Dialogue Schema Representation

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 schema¹, which is followed by sections containing Episodic Logic (EL) formulas. EL is a type-coherent, expressive intensional logical form designed to closely resemble the surface form of English; the syntax and semantics of EL are omitted from this proposal for brevity, but the reader is referred to (Schubert and Hwang, 2000) for these details. The sections that a dialogue schema may contain are as follows²:

- 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-fluent³ type predications 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 predications about individuals occurring in the schema apart from the types, e.g., that **?x** is yellow.

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

 $^{^2 {\}rm The}$ section names are slightly modified from, but mostly analogous to, those discussed in (Lawley et al., 2019)

³The terms "fluent" and "non-fluent" are borrowed from situation calculus to refer to predicates whose extensions are subject to change over time and those that remain fixed over time, respectively.

- **static-conds**: fluent predications which are expected to hold throughout the schema episode, e.g., that the hearer is located near the speaker.
- **pre-conds**: fluent predications that are expected/required to be true at the initiation of the episode represented by the schema.
- **post-conds**: fluent predications that are expected/required to be true at the conclusion of the episode represented by the schema.
- **trigger-conds**: fluent predications 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 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 3.1, taken from a "concept tutoring" task in the Blocks World domain (discussed more in Chapter 4). Variable symbols in this example schema take on three different prefixes: '^', '?', and '!'. The first of these represents indexical variables, whose values point to some object within the context (for example, **`you**, **`me**, or **`now**). '?' and '!' are used to represent individual variables and sentential variables, respectively. Furthermore, 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. (<formula> ** ?e2).

The example schema showcases various types of episodes that may appear in a dialogue schema. Some of the episodes, such as '?e1', correspond to primitive actions that can be directly executed by the agent, while others such as '?e3' correspond to complex actions that trigger the activation of a sub-schema for that action. Episodes

```
(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 add-to-context.v (that (^you understand.v ?c)))
     ?e9 (^me say-bye.v))
    ; If user doesn't understand concept, try building another example.
    (:else [...] ))
  [...]
)
:certainties (
  !c1 (!e5 0.4)))
```

Figure 3.1: Example of a (simplified) dialogue schema, from the Blocks World concept tutoring task discussed in Chapter 4.

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.

Schema Library Semantic Interpretation Active Schema **Perceptual Subsystems** :header ((^me ask-about-prognosis ^you) Gist-clause ULF context **(**)) Interpretation Interpretation ?g1 (`^me want.v (that (^me know.v ...))) episodes (Reference Resolution ?e1 ^me sav-to.v ^vou '(...) direct observation Fact Store schema selection Context **Hierarchical Planner** goal replanning Curr Plan Memory E1 (^me say-to.v ^you '(...)) focus attention Knowledge update facts ?e2 (^you reply-to.v E1) Base ?e3 (^me react-to.v ?e2) Response Generation Specialist Subsystems Gist-to-Response T 品 if system if expected action . event ULF-to-Gist consult Execute Action Match Expectation specialist eaction Selection update variable bindings activate schema for react with new schema non-primitive action

3.2 System Architecture

Figure 3.2: A diagram of the architecture of the Eta dialogue manager.

A diagram of the overall architecture of Eta is shown in Figure 3.2. The dialogue manager is built around 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. The dialogue manager also has access to a

schema library containing schemas that can be instantiated and used to construct hierarchical plans. These plans are used to guide the dialogue, though they are flexible and can be modified or abandoned "on the fly" when a new schema is activated. The example plan in Figure 3.2 shows the system at a state where the system utterance had just been executed (indicated by an arrowhead), and the subsequent user reply is expected.

Eta is designed to interface with external subsystems of two types: **perceptual subsystems** that provide perceptual knowledge (e.g., visual or auditory information) to Eta, and **specialist subsystems** that are capable of specialized, domain-specific reasoning (e.g., spatial planning, mathematical reasoning, etc.). When appropriate, Eta will use a **semantic interpretation** pipeline to interpret the meaning of observations in light of the dialogue context. Finally, if warranted by the dialogue plan, Eta will use a **response generation** module to generate English responses, or possibly react with some schema.

Each of these components are described in greater detail in the following sections. Several of these components are processed independently of each other through the help of an indefinitely looping **task queue**, which is described in Section 3.3.

3.3 Dialogue Tasks

Humans, in dialogue, are able to manage multiple capabilities asynchronously – for instance, we can observe our surroundings at the same time we interpret our interlocutor's speech and plan our next reply. While the Eta system is not strictly asynchronous, it simulates asynchronous processing through the use of a *task queue*. This queue allows for the interleaving of various processes such as plan execution, observation, and inference – each is executed for a short period of time before the next task is chosen. Specifically, the task queue repeatedly cycles between the tasks described below (each one corresponding to one or more components in Figure 3.2):

- 1. perform-next-step: handles the currently pending step in the dialogue, performing each of the functions described in Sections 3.7 and 3.8 (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 Section 3.6. 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 may select new candidates for the active schema, and/or instantiate a candidate as the active schema via schema matching (the new active schema may replace the old one completely, or form a subplan), though this remains future work see Chapter 5.

3.4 Perceptual and Specialist Subsystems

Eta is designed to be able to connect with arbitrary subsystems that act as modules for perception or specialist reasoning. The specific subsystems that Eta interacts with may be configured on a per-avatar basis. Each subsystem listed in the system's configuration corresponds to a single input and output file, and Unscoped Logical Form (ULF – a variant of EL with aspects such as scoping and tense left unresolved) queries are used as a medium of communication for each.

The most commonly used perceptual subsystems are audio (via an automatic speech recognition software) and text input, but the Blocks World avatar in Chapter 4 also makes use of a vision subsystem, which supplies Eta with object coordinates and observed physical actions (e.g., block moves). If warranted by the dialogue plan, it is possible for Eta to send a perceptual subsystem a request for particular information or to focus attention on a particular observation.

Some examples of specialist subsystems used by Eta are as follows. A temporal reasoning module allows the system to backtrack through time from a particular episode – reconstructing past world states while applying temporal constraints (e.g., from adverbials); for more details on this module see (Kane et al., 2020). A spatial reasoning module allows the system to evaluate spatial relations between objects using computational models of spatial relations (Platonov and Schubert, 2018). The system is also able to draw from taxonomic resources such as a noun hierarchy, with specialized routines for making taxonomic comparisons.

3.5 Facts Store

Each of Eta's components interact closely with a central fact store that maintains various facts (in EL form) that the system is assumed to know – including general world knowledge, stored in the *knowledge base*, episodic memory of past events, stored in *memory*, and facts that are assumed to be in the dialogue's common ground (i.e., "true now" and reflexively believed by both participants), stored in *context*.

When a new fact - e.g., ('you say-to.v `me '(...)) - is added to context, the system will create a new episode variable to be (partially) characterized by the fact, as well as creating a timestamp marking the earliest time at which the fact is known to be true. The following atemporal facts would thus be added to memory for this example: (('you say-to.v `me '(...)) * E3), (|Now14| during.p E3), (NOW* during E3), where NOW* is an indexical variable that evaluates to the current time. When a fact in context is observed to be no longer true, the fact is removed from context and the indexical NOW* predicate in memory is updated to an upper bound timestamp, e.g., (|Now17| during.p E3). For a small subset of telic predicates, such as say-to.v, the system continually monitors context for the truth of such facts and automatically removes them after a set number of task queue cycles.

3.6 Semantic Interpretation

As the system receives inputs from perceptual subsystems, these facts are added directly to the fact store, 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 levels, the assumption is made that to effectively "ground" the user's utterance and resolve ellipsis (as well as many common cases 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. For instance, the utterance "Thai is nice" might mean many things in different contexts, but within the context of the question "What type of foods do you like?", the possible meaning of the utterance is narrowed down to an indication of the user's preferred cuisine.

The first level, gist-clause interpretation, attempts to extract an explicit, contextindependent "natural language" sentence representing the *gist* of the user's reply. The motivation behind gist-clause interpretation is that if relatively superficial text processing can be used to initially map complex utterances to simplified contextindependent clauses, any downstream semantic interpretation and response genera-



Figure 3.3: An example pattern transduction tree for gist clauses, with a trace for specific example inputs shown. The context of the previous Eta utterance is first used to select an appropriate topical tree for interpreting the user's utterance, which is then matched to arrive at a final gist clause.

tion are greatly simplified. A hand-engineered ULF parser, for instance, would only need to consider inputs in a restricted format rather than unbounded inputs.

In practice, gist-clause interpretation is done using the *hierarchical pattern transduction tree* method previously introduced by 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 (prefixed by a period), 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 3.3.

After extracting gist-clauses, Eta also can derive an unscoped logical form (ULF)



Figure 3.4: 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.

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 EL. 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 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 word sequences to transduction hierarchies appropriate for their phrasal types; these recursively derive and 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 transduction tree used for parsing a historical question into ULF is shown and described in Figure 3.4. 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.

3.7 Hierarchical Planner

The hierarchical planner component maintains a pointer to the current dialogue plan, which is represented as a graph of plan steps where each step has a pointer to the previous step, next step, and a subplan (when applicable).

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 corresponding 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 $\lim_{c\to 1} T(c) = \infty$, as desired. In the case where c = 1, the system will halt indefinitely for the expected input.

If the currently pending step is a system action, the system will *execute the action*. If the action is non-primitive, it will select the appropriate subschema and expand it as a subplan to the current step, 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 function for that action: it might update the fact store; 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 the copy of the schema that the plan is instantiated from.

Plans can be modified over the course of the dialogue depending on conditions defined in the respective schema. In the current version of the system, the goal conditions of a schema may modify the dialogue plan in two respects: first, if an episode instantiates a schema whose goal is deemed to be already satisfied by facts in context, 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 simply moving on).

3.8 Response Generation

The system's response generation is governed, at the highest level, by a system reaction selection module – this 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 semantic interpretations of the user, response generation may require interaction with specialist subsystems and the system's fact store to generate reactions, as well as requiring other syntactic transformations prior to conversion to surface-form utterances. The process of generating a response ULF is currently handled by domain specific modules, such as the historical question-answering module described in (Kane et al., 2020).

However, once a 'final' response ULF is obtained, the process of mapping it to a surface-form utterance is fairly straightforward (due to the close correspondence between ULF and natural English; a distinct advantage of ULF over other more opaque semantic representations), and follows a two-stage process resembling the process for interpretation. First, the response ULF is mapped to a simple contextindependent response gist-clause⁴. Second, this response gist-clause is mapped to an actual response utterance using hierarchical transduction trees – given the context of the previous dialogue turn – before being output by the system.

⁴This mapping is done using the ULF2English library: https://github.com/genelkim/ulf2english

Chapter 4 Applications of Eta and Results

The capabilities of the Eta dialogue manager are put to the test when applied in two practical domains, both requiring challenging planning and reasoning capabilities. The first application of Eta that I discuss in this chapter 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 (about both spatial relations among blocks as well as historical states of the world), concept tutoring, and (as an anticipated future goal) concept learning. The second application of Eta is the SOPHIE (Standardized Online Patient for Healthcare Interaction Education) avatar, a virtual agent that can act as a patient to help medical students practice having difficult conversations with patients. Such conversations require deep understanding and mixed-initiative interaction, making them a challenging dialogue domain.

4.1 Blocks World Avatar (David)

4.1.1 Domain and Experimental Setup

The "blocks world" domain has a rich history in AI research (beginning with the simulated SHRDLU collaborative planning system (Winograd, 1972)) due to the deep semantic understanding and spatial reasoning required to complete tasks within the domain – ideally, a system that is competent in a simplified blocks world can scale up to a system that is competent in more general domains, such as a "room world" with everyday objects.

The David virtual agent is intended to hold collaborative dialogues with a user within a physical blocks world setting, depicted in Figure 4.1a. The blocks on the table can be referred to by associated company names (e.g., "the Twitter block") 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 4.1b, builds upon the core Eta architecture in a fairly straightforward 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.). In contrast with past work in the blocks world domain, this specialist subsystem supplies Eta with the ability to use fully general mathematical models of spatial relations (such as "near" or "behind") that can ultimately be re-used in more realistic domains (Platonov and Schubert, 2018; Platonov et al., 2021b).



Figure 4.1: The blocks world apparatus setup (a), and dialogue pipeline (b). In (b), the arrows indicate the direction of interaction between the modules.

4.1.2 Question-Answering Task

The David virtual agent was initially configured for an interactive question-answering task, with the aim of developing the reasoning capabilities necessary for full collaborative planning. In this task, users are able to move blocks around on the table while asking the agent free-form questions about the world. These questions may include *spatial questions* about relations or properties of the blocks on the table, *historical questions* about past world states or actions, and *explanation questions* asking the system to explain an answer that it gave.

The overall top-level dialogue schema used for this task is shown in Figure 4.2. The main content of the schema is a repeating episode wherein the user is expected to reply to the system with a question, a special request (e.g., to pause the conversation), or smalltalk (e.g., asking for the system's name), and the system reacts accordingly. In the case where a spatial or historical question is asked, the system reacts with a

```
(event-schema : header (((set-of ^me ^you) have-QA-dialogue.v) ** ?e)
[...]
:episodes (
 ; David introduces himself.
 ?e1 (^me say-to.v ^you
    '(Hi, I'm David. I'm ready to answer your spatial questions.))
  ; Repeat prompting the user for a spatial question until finished.
  ?e2 (:repeat-until (?e2 finished2.a)
   ; Prompt the user for a spatial question.
   ?e3 (^me say-to.v ^you
          '(Do you have a spatial question for me?))
    ; User replies with spatial question, special request, or smalltalk.
   ?e4 (^you reply-to.v ?e3)
   ?e5 (:try-in-sequence
      ; If user says goodbye or asks to pause, react accordingly.
      (:if (^you say-bye.v)
       ?e6 (^me add-to-context.v (that (?e2 finished2.a)))
       ?e7 (^me react-to.v ?e4))
      (:if (`you paraphrase-to.v `me '(Pause for a moment .))
       ?e8 (^me react-to.v ?e4)
       ?e9 ((set-of ^me ^you) pause-conversation.v))
      ; Otherwise, react to spatial question (or smalltalk).
      (:else
       ?e10 (^me react-to.v ?e4))))))
```

Figure 4.2: The main dialogue schema used for the blocks world question-answering task.

subschema that respectively consults the spatial specialist or the temporal specialist to generate a response.

Spatial Questions

Spatial questions may include questions about the spatial relations between blocks, positions of blocks on the table, physical properties of blocks (e.g., color). Some examples of questions answerable by the system are as follows:

- "What color is the leftmost block?"
- "Which blocks are touching some red block?"

- "Where is the Texaco block?"
- "Is the block between the Twitter block and Texaco block clear?"
- "How many red blocks are below a blue block?"

In each case, the system attempts to derive a ULF interpretation of the question (after some gist-clause preprocessing). For example, the system might obtain ((the.d (most-n left.a block.n)) ((pres be.v) (of.p (what.d color.n)))) from the first question. The interpretation is sent as a ULF query to the spatial reasoning specialist, which uses a constraint solver to produce a list of answer relations with confidence scores (Platonov et al., 2020). Eta's response generator attempts to substitute these answer relations for the corresponding wh-noun-phrase in the question ULF before converting to English – potentially using confidence scores to filter answers or insert uncertainty adverbs such as 'possibly'.

Historical Questions

Historical questions may entail any spatial question combined with temporal adverbials (e.g., "Which blocks were near a red block in the beginning?"), questions about past actions on blocks (e.g., "What block have I moved three times?"), or possibly even complex questions involving subordinating conjunctions or relative clauses (e.g., "Where was the Twitter block before I moved it?" or "What blocks are below the block that I just moved?").

Based on practical and cognitive constraints on spatial memory (Rensink, 2001), we avoid maintaining a detailed visual or vector-based memory, but rather assume that Eta has access to a high-level symbolic memory of past block moves with which the agent can reconstruct past scenes, given currently observed block locations. When Eta detects a historical question, it consults a temporal specialist capable of using this symbolic memory to backtrack through time, reconstructing the approximate world state at each episode, all while applying temporal constraints from the question ULF (e.g., from temporal adverbials) to obtain a list of answer relations. For more details on this algorithm, see (Kane et al., 2020). Eta generates a response from these answer relations in the same manner as with spatial questions.

Explanation Questions

To further test David's spatial understanding, we extend the spatial question-answering task by allowing the user to request explanations of the answers that David provides – for instance, if the user asks "Is the Twitter block on top of the Texaco block?" and the system replies "Yes", the user might follow up with "Why is it on top of the

```
DAVID: Would you like to ask me a spatial question?
User:
      (moves a block)
DAVID: 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.
      When did I move the Toyota block?
User:
DAVID: You moved the Toyota block three minutes ago.
```

Figure 4.3: Example of the sort of dialogue that David can engage in (in the QA setting).

Texaco block?". If David answers with "Because the Twitter block is above the Texaco block and touching it", the user might again follow up with "Why is the Twitter block above the Texaco block?", to which the system would reply with "Because the Twitter block is higher than the Texaco block based on the positions of their centers".

Interpretation of explanation requests is handled in the same manner as ordinary spatial questions. The spatial reasoning specialist uses a factor graph representation to produce low-level relations that justify the relation the user is inquiring about, which Eta then verbalizes in a "natural" manner (Platonov et al., 2021a).

Evaluation Results

The performance of the agent in answering both types of questions was evaluated in (Platonov et al., 2020) and (Kane et al., 2020), respectively. Five volunteers were enlisted (including native and nonnative English speakers) and instructed to move the blocks around and ask questions in each category, with no restrictions on wording¹. After the agent displayed its answer, participants were asked to provide feedback on the quality of the answer by marking it as correct, partially correct, or incorrect.

¹Due to the COVID-19 pandemic, the historical question evaluation was conducted using a simulated system with text-based interaction.

Results for spatial questions and historical questions are shown in Tables 4.1 and 4.2. The parser accuracy is evaluated by examining the number of well-formed questions that result in semantically correct ULF expressions. In the case of the spatial question evaluation, the rate of incorrect parses that are attributed to automatic speech recognition errors is also shown. Overall, the semantic parser itself is highly reliable, with 94% of grammatical sentences being parsed correctly in the text-based historical question evaluation. The QA results are also encouraging, with the system giving the correct answer in a majority of the cases for both question categories. An example of a transcript from the David avatar in the QA task is shown in Figure 4.3.

Table 4.1:	Spatial	question	results.
------------	---------	----------	----------

Total well-formed
questions329Parser accuracy74%% Incorrect parses
due to ASR errors53%Correct answers66.6%Partially correct13.7%Incorrect18.8%

 Table 4.2: Historical question results.

Total well-formed	387
questions	
Parser accuracy	94%
% Incorrect parses	N/A
due to ASR errors	
Correct answers	77%
Partially correct	3%
Incorrect	20%

A quantitative evaluation of the system's performance in answering explanation questions remains future work.

4.1.3 Concept Tutoring

The concept tutoring task attempts to enable interactive collaboration between the David avatar and a user. In this task, David has in mind some abstract concept (e.g., an arch), and attempts to interactively teach the user the concept through having them create examples of the concept with blocks.

A simplified version of the dialogue schema used for concept tutoring is shown in Figure 3.1 (in which it is used as an example for illustrating the general dialogue schema representation). The agent has access to a set of structure concepts, which are themselves represented as object schemas – schemas that represent stereotypical properties of object concepts (e.g., their spatial relations and parts structure). The agent randomly chooses a concept that the user doesn't understand, specializes the schema for that concept into a simple example schema involving the blocks on the table, and guides the user through step-by-step construction of that example. The object schema specialization, as well as construction of a step-by-step plan to realize the object schema, are handled by the spatial specialist subsystem.

```
DAVID: I would like to teach you the concept of an arch.
User: Okay, let's go ahead.
DAVID: Let's start with a simple example. Put a block on the table.
User: Sure. (places block A centrally)
DAVID: Now put a block to the right of block A.
User: (places block B touching the right side of block A)
DAVID: Well, block A should not touch block B.
User: Okay. (shifts block B half a block to the right)
DAVID: Good. Now place a block on top of block A and block B.
User: (places block C on top of both blocks)
DAVID: Excellent. You have now built the simplest possible arch.
DAVID: Do you think you understand the concept?
User: Can we try making a bigger one?
DAVID: We certainly can. Why don't you try building it?
User: (creates two 2-tall columns with blocks A-D)
User: How's that?
DAVID: Good, that's correct so far.
User: Should block E be on top of block C and block D?
DAVID: Yes.
User: Okay. (places block E on top of block C and block D)
DAVID: Looks like the structure is completed.
DAVID: I think you've got the idea of an arch. You caught on fast!
```

Figure 4.4: Example dialogue that David can hold from the concept tutoring task. The company block identifiers are replaced with letters for brevity.

If the user still doesn't understand the concept after building an example, the agent may direct them to create another example of the structure (e.g., "Can we make a bigger one?"). The agent will then modify the object schema with the given constraint (e.g., a bigger instance of the concept schema) and supervise the user through construction of the new example – issuing corrections if the user places a block in the wrong location, until the goal structure is met.

Carrying out a quantitative evaluation of this task remains future work – this will likely involve collecting annotations of a number of concept tutoring dialogue transcripts, as well as measuring efficacy of the system in teaching spatial concepts to participants (where spatial concepts such as "arch" would be replaced by madeup names). However, a transcript from a demonstration of the system is shown in Figure 4.4.

Note that throughout this process, the system retains its ability to answer spatial and historical questions, as well as being able to answer modal questions about the step-by-step plan – as shown in the turn where the user asks David "Should block E be on top of block C and block D?". Adding this capability required only

including the question-answering schema as a subschema in the concept-tutoring schema, illustrating the modularity of our approach.

4.2 Discussion

To summarize our results from the blocks world domain, we found that a handengineered gist-clause to ULF transduction tree makes for a highly accurate domainspecific semantic parser, while dialogue schemas provide a flexible and modular representation for enabling collaborative dialogues between an agent and a user. Furthermore, the system is able to effectively use specialist subsystems to accurately reason about spatial and temporal relationships, as demonstrated in the respective QA tasks.

However, our system also runs into several limitations. First, the nature of the semantic parser design means it's limited to the forms of natural language that are encountered in the restricted blocks world setting – in more realistic settings, such as the ones modelled by Platonov et al. (2021b), a wider variety of nouns and phrasal constructs may be encountered. A more general approach to ULF parsing is required in this case.

Second, due to the lack of an algorithm for monitoring and activating possible alternative dialogue schemas, the system has no ability to handle unexpected inputs – for example, if the user were to say "Can we build a different structure?" in the concept tutoring dialogues, ideally the system should trigger a dialogue schema for choosing a new concept rather than carrying on with its current structure-building schema.

Finally, the temporal reasoning specialist used in the historical QA task (described in (Kane et al., 2020)) is fairly simplistic – relying on backtracking through a linear chain of episodes while applying temporal adverbials as constraints. In the general case, temporal relations between episodes do not form a single linear chain, but rather a directed acyclic graph (Miller and Schubert, 1990). The semantics of some temporal adverbials – particularly habituals such as "always" – are also considerably more complex than how these adverbials were handled in the historical QA task. These issues are revisited in Chapter 5.

4.3 Standardized Patient Avatar (SOPHIE)

4.3.1 Domain and Experimental Setup

The SOPHIE (Standardized Online Patient for Healthcare Interaction Education) avatar is a multidisciplinary project – carried out in collaboration with URMC and the ROC HCI lab – to create a standardized virtual patient that doctors can use to practice patient communication while receiving dynamic feedback. Communication skills on the part of the doctor are a well-recognized determinant of patient satisfaction and optimal outcomes, yet communication training is a resource-intensive process – requiring extensive training and taxing emotional investment from patient actors. As a result, the quality and availability of communication training are often quite variable. The ability to create and customize virtual conversational patients that are capable of having realistic conversations with doctors could have a broad impact on medical training.

Moreover, this application domain also provides opportunities to improve the underlying Eta dialogue manager. The types of conversations that doctors have with patients are diverse and often involve complex mixed-initiative dialogue – i.e., a mixture of the patient taking initiative and the doctor taking initiative. These types of domains are still very much an open area of dialogue research, as they demand much more sophisticated dialogue management and planning techniques than single-initiative dialogues (Walker and Whittaker, 1990). While some virtual patient systems already exist (Maicher et al., 2017; Carnell et al., 2015; Rossen et al., 2009), they address simpler single-initiative interactions such as history-taking (where the doctor asks the patient a series of questions and the patient gives factual replies). To our knowledge, there is no virtual patient system capable of mimicking the complex conversations that occur when, for instance, a doctor has to break bad news to a patient.

In its current state, the SOPHIE avatar is a virtual patient who has been recently diagnosed with lung cancer and is seeking medical advice, allowing medical students and oncologists to practice having difficult conversations with patients (Ali et al., 2021). The Eta dialogue manager is used to interpret the user (after collecting inputs from an automatic speech recognition subsystem) and generate appropriate responses. In a similar vein as the LISSA social skills practice system, 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.

4.3.2 Dialogue Design

Dialogue management for the SOPHIE avatar largely relies on the level of shallow gist-clause interpretation and reaction generation described in Sections 3.6 and 3.8. Although this level of interpretation doesn't afford "true understanding", it is sufficient for most of the types of responses that SOPHIE may come across – most inputs can be reduced to a simple, explicit form given the context of the immediately preceding utterance. If the doctor makes a statement in response to SOPHIE (i.e., SOPHIE has the initiative), the system uses SOPHIE's previous question to choose an appropriate topical gist-clause tree for interpreting the user's statement. If this fails, or if the doctor asks a question to SOPHIE (i.e., the doctor has the initiative), the system will use a general question gist-clause tree to interpret the user's input. If the input was recognized as a question but no gist-clause was matched, the system will make a clarification request – however, the system will only ask the user to clarify a couple times before proceeding with its plan.

In turn, SOPHIE can react to a gist-clause by either making a statement to the user (e.g., "*That's really hard to hear...*"), or activating a new schema for an appropriate subdialogue (e.g., "*What should I tell my family?*"). Schemas for subdialogues typically consist of a single "adjacency pair" with an agent speech act followed by a user speech act, sometimes with an explicit goal (for instance, to know how to inform her family) defined. This allows for the flexibility required for mixed-initiative dialogue, as any subdialogue schema can be interrupted by another when necessary.

Furthermore, the SOPHIE agent has the ability to use schema goals to replan dialogue in particular circumstances – for example, if a goal is not satisfied upon the conclusion of the schema, a transduction tree may be used to select some appropriate follow-up behavior (if any; by default the system simply moves on with the current plan). On the other hand, the agent may skip over a schema if the goal of that schema is already satisfied, such as if the doctor had indirectly answered the agent's question earlier in the conversation.

The pattern transduction rules were meticulously designed by myself and a research assistant, using three sources of data to anticipate possible user inputs and optimal patient reactions:

- 1. The VOICE dataset (Hoerger et al., 2013): A corpus of human-human conversation transcripts from a previous patient actor study, consisting of 109,134 dialogue turns across 389 conversations (a subset of which were selected for development based on relevance to SOPHIE's background).
- 2. Guidance from palliative care experts on common behavior from doctors and patients.

3. Feedback and transcripts from initial users of the system in a small focus group study (Ali et al., 2021).

The avatar's top-level dialogue schema and pattern transduction rules are designed to steer the conversation towards the doctor getting the patient to understand her test results, prognosis, what her condition means for her future, and how to break the news to her family (typically in that order, though the system is fairly flexible). However, due to the modular nature of the schema-based system design, creating additional dialogue modules from the existing rules – such as a dialogue pertaining to treatment options – is fairly trivial.

4.3.3 Pilot User Experiment

Methodology



Figure 4.5: The design of the SOPHIE pilot experiment.

In order to evaluate whether the SOPHIE system can help doctors improve aspects of patient communication, a randomized pilot study was carried out. An overview of the experimental setup is shown in Figure 4.5. We selected 30 participants with medical backgrounds and randomly divided them into a treatment and control group, balancing for level of experience. The treatment group underwent the educational intervention with the SOPHIE system while the control group received no training. Communication performance was evaluated using URMC-affiliated human standardized patient (SP) actors over Zoom; after the conversation, the SP would rate the conversation using a standard scale. The ratings for the treatment group were compared with the control group to see if there were any statistically significant differences.

Results

We found that for every question in the SP rating scale, the treatment group performed better than the control, although not all of the questions were statistically significant with a p-value of 0.05. Notably, the treatment group had a statistically significant gain on the "overall communicator" score. Several other questions on the rating scale where the treatment group performed significantly better are as follows:

- The participant asked how much information the patient would like concerning prognosis.
- The participant gave the SP many opportunities to talk.
- The participant empowered the SP.
- The participant was generally empathetic.
- The participant validated the SP's emotional responses.

These results suggest that the dialogue manager, in conjunction with the feedback module developed by my ROC HCI collaborators, provides the doctor with opportunities to tangibly improve their communication skills through practice.

Transcripts of SOPHIE interactions from the patients in the treatment group were collected and used for further dialogue analysis, described in 4.3.4. Some "cherry-picked" dialogue excerpts from the pilot transcripts are shown in Figure 4.6.

4.3.4 Dialogue Evaluation

Methodology

In order to evaluate the quality of the SOPHIE pilot dialogues, we collect turnby-turn human annotations on the system's responses given local dialogue context. This method of evaluation has previously been employed in virtual patient research, with past systems reporting between 60% and 80% response accuracy (Maicher et al., 2017). However, as we noted in Section 4.3.1, these systems were evaluated in

SOPHIE:	I've already scanned the results, but don't understand what
	they are telling me. Could you explain what they mean?
User:	So unfortunately Sophie I have some bad news. It looks like the
	cancer has grown and spread.
SOPHIE:	Those are not the words I wanted to hear. I mean, I was bracing
	for the worst, since I could tell by the pain that it's bad.
	But to hear that the cancer has spread is quite depressing.
	What does it all mean for me?
User:	How much information would you like to know about the prognosis?
SOPHIE:	I feel very anxious about my condition. I feel like it's gotten
	worse. I want to know what this means for me.
User:	What concerns do you have about the future?
SOPHIE:	What worries me the most is just how uncertain my future is.
	Some days I believe my condition will get better and others
	I brace myself for the worst. Is there any way you can let me
	know what I should expect?

(a)

SOPHIE:	I still don't understand how this happened! Just three months
	ago I didn't even know I had cancer.
User:	I agree that lung cancer in particular is a very difficult
	diagnosis for patients and their families.
SOPHIE:	Cancer is such a challenging disease to live with. But I
	appreciate all you've been doing to help me.
SOPHIE:	I haven't told my family everything yet. I wanted to wait to
	talk to you first. What should I say to them?
User:	How much information do you want them to know?
SOPHIE:	My family knows about my cancer already, but they don't
	really know how bad it is, or what it means for me. How should
	I discuss these with them?
User:	Being open and honest is going to help you the most, and if
	you'd like they can come to a visit with you and we can help
	you talk to them.
SOPHIE:	Okay. It will be hard to give them such bleak news, but they
	deserve to know the truth.

(b)

Figure 4.6: Some "cherry-picked" dialogue excerpts from the SOPHIE pilot study.

simpler single-initiative domains – where the system merely has to select one of a set of responses that best answers a question from the doctor – and thus are not directly comparable to our system.

Expert Annotations From the pilot experiment, we obtained 397 dialogue turns across 30 dialogues². As a preliminary analysis, myself and another RA involved in the SOPHIE project independently provided expert annotations on whether the system extracted a correct gist-clause, an incorrect gist-clause, or failed to extract a gist-clause, as well as whether the system gave an appropriate response, an inappropriate response, or asked the user to clarify. We also annotated any significant ASR errors that were observed, which tended to fall into two categories: transcription errors and turn-taking errors (i.e., where the ASR cut the user short).

Neural Baseline We also attempt to establish a "neural baseline" for our conversation domain – that is, the performance of a state-of-the-art language model fine-tuned on standardized patient transcripts – and compare the responses generated by this model on the pilot data to our system. Specifically, we fine-tuned DialoGPT-medium (Zhang et al., 2020) on the VOICE standardized patient dataset (filtering for only patient turns, with a context window size of 5) for 5 epochs, with a batch size of 1. This resulted in a validation set perplexity of 6.53; gains after 5 epochs were negligible. To generate model responses for the pilot data, for each turn in the dataset, we concatenated the user utterance with the context of the immediately preceding dialogue turn, separated by end-of-turn tokens, and let the model generate the next response. We used a length penalty of 0.5 and a repetition penalty of 1.4 in generation, as these were found to generate the highest quality responses through manual experimentation.

Crowdsourced Annotations We then crowdsourced annotations on response quality from Amazon Mechanical Turk. We removed 89 items that were judged by either of the expert annotators to include significant ASR errors in the doctor's input, resulting in 308 items – each consisting of the context of the previous patient utterance, the doctor's utterance, the response generated by Eta, and the response generated by the neural baseline model. Items were randomly distributed into 20 Human Intelligence Tasks (HITs), each containing 16 items plus 2 "sanity-check" items (i.e., hand-crafted responses designed to unambiguously elicit particular ratings, for use in assessing the noise in the annotations). To avoid introducing unintended annotation bias, items were exactly balanced on string length of each text field (i.e.,

 $^{^{2}15}$ participants in the treatment group used the system for two sessions each.

four high/low bins for each text field computed using a median split), and were approximately balanced on the expert annotations of response quality (to ensure that, for instance, "Could you repeat that?" responses are evenly distributed across HITs).

For each item, workers were shown a response from Patient A to the previous two dialogue turns, and then asked to rate the following four questions about the response on a Likert scale ("Strongly Agree" to "Strongly Disagree"). Then workers were shown a response from Patient B to the same previous turns, and asked to rate the same four questions. The Eta responses and neural baseline responses were randomly assigned to Patient A and Patient B.

- Q1. Patient A/B's response is fluent and natural.
- Q2. Patient A/B's response is consistent with her having understood the preceding turns.
- Q3. Patient A/B's response is consistent with her role as a cancer patient.
- Q4. Patient A/B's response expresses appropriate emotions (if an emotional response).

Workers were instructed to focus on the quality of the response irrespective of the patient background and previous turns for answering Q1, and to overlook any slight fluency issues in the response for answering Q2-Q4. Additionally, Q3-4 had "Not Applicable" options, in the event that these questions could not be evaluated (for instance, Q3 is impossible to evaluate if the patient asks "Could you repeat that?", and Q4 is impossible to evaluate if the patient says "I'm taking Lortab for the pain.").

Results

Table 4.3 shows the averaged expert annotations (N=2) for ASR errors, the fraction of correctly extracted gist-clauses, and the fraction of appropriate responses from the system (the latter quantity is expected to be higher than the former because the system might fail to extract a gist-clause but still give an appropriate "default" response). These statistics are corrected for ASR errors by assuming that a failure to extract a gist clause, and thus a clarification request by the system, are the correct behaviors when the input contains a significant ASR error. We also show the fraction of responses that are judged as appropriate conditioned on a gist-clause having been extracted. Interannotator agreement (Cohen's kappa) was quite high for both annotations, at 0.85 for gist-clause annotations and 0.71 for response annotations.

From the crowdsourced Amazon Turk study, we obtained 10 ratings for questions Q1-4 for each item, from 167 native English speakers. The resulting distributions of

ASR Errors	20%
Correct Gist Extracted	39%
No Gist Extracted	41%
Incorrect Gist Extracted	20%
Appropriate Response	49%
Clarification Request Response	28%
Inappropriate Response	24%
Appropriate Response Given Gist Clause Extracted	72 %

 Table 4.3: Results from the preliminary expert annotations.

ratings per item (averaged across participants, discarding 'N/A' values) are shown as histograms in Figure 4.7 (a score of 1 corresponds to "Strongly Disagree", and 5 to "Strongly Agree"). The mean and median scores for each question across all items are shown in Table 4.4; the mean differences between Eta and DialoGPT were found to be statistically significant for each question (P < 0.05) using a Mann–Whitney U test.

Agent	Eta		DialoGPT		Difference	
Question	Mean	Median	Mean	Median	Mean	Median
Q1	4.15	4.20	3.49	3.55	+0.66	+0.65
Q2	3.29	3.20	3.03	3.00	+0.26	+0.20
Q3	3.78	4.00	3.30	3.40	+0.48	+0.60
Q4	3.60	3.60	3.06	3.11	+0.54	+0.49

Table 4.4: Mean and median of average responses across items for each set of responses.

4.4 Discussion

While the expert-annotated metrics shown in Table 4.3 are fairly low compared to the standardized patients described in (Maicher et al., 2017), we argue that these results are consistent with the more complex conversation domain than the one those systems were developed for. To our knowledge, there is no standardized patient system deployed in a mixed-initiative conversational domain similar to ours, so we view these results as a useful baseline for future work.

A majority of failures to extract the correct gist-clause were due to failures to extract a gist-clause at all, rather than incorrect gist-clauses. Interestingly, in the

Figure 4.7: Distributions of average crowdsourced response ratings for each question.

cases where the system was able to extract a gist-clause, the fraction of appropriate responses by the system was significantly higher – suggesting that the main bottleneck facing the system at the moment is the gist-clause extraction step. In other words, the system performs quite well when it is able to extract a gist-clause, but flounders when it is unable to extract a gist-clause (in the pilot, this would typically cause the system to make a clarification request – this happened frequently and would sometimes cause the user to become frustrated).

Concerning the crowdsourced ratings shown in 4.4, the Eta dialogue manager was rated higher on average for each question than the fine-tuned DialoGPT model, with the largest gains being in fluency/naturalness, consistency with the role of a patient, and appropriateness of emotions. We take these results to be an indication that the customizability of Eta provides a distinct advantage over an end-to-end deep learning approach, since our system allows the interpretive rules and responses to be designed with a particular role and character in mind. In contrast, the fine-tuned DialoGPT model would sometimes get its role in the dialogue mixed up (e.g., act as a doctor rather than a patient), an issue related to the well-known tendencies of end-toend neural dialogue models to contradict themselves and hallucinate false knowledge (Roller et al., 2020). This is critical because it only takes a single out-of-character response to disrupt the user's immersion in a conversation.

The lowest ratings for both sets of responses – and the smallest gains by Eta above the neural baseline – were concerning whether the agent demonstrated understanding of the previous turns. This illustrates that the superficial pattern-matching rules used by Eta are not, at least in their current form, sufficient to demonstrate understanding. An interesting phenomenon that can be observed in Figure 4.7 is that the distribution of ratings for Eta on this question is strongly bimodal, unlike DialoGPT. We believe that this pattern attests to the results found in the expert annotations, namely, that the *conditional* probability of Eta generating a good response given that it extracted a gist-clause is high, but when the system fails to extract a gist-clause, it generally gives poor responses (typically a clarification request such as "I'm sorry, I didn't quite understand. Could you repeat that?"). The average rating for this question, then, could likely be greatly improved by increasing the coverage of gist-clause extraction.

Finally, we note one limitation of this evaluation method. The annotations collected only take into account "local" context, i.e., whether a response is appropriate given the immediately preceding turn. It doesn't capture "global" coherence in the dialogue – some issues that we noticed in practice were a tendency to repeat certain generic responses (e.g., "I suppose these things can be hard to predict."), and occasionally contradicting itself (e.g., "What about chemotherapy?", followed by "I'm not ready to talk about treatment options..." a few turns later). While some of these issues can be easily resolved through modification of the pattern transduction rules, ultimately the issue of global coherence requires the system to be able to store and make use of facts relating to the user and previous conversation.

Despite these issues, we observed from the pilot experiment that users were still able to meaningfully interact with the system to improve their patient conversation skills – we believe that even in cases where the system doesn't understand the user very well, the opportunity for a doctor to practice giving responses to a virtual patient is valuable, with future improvements in the dialogue manager only making this experience more realistic. In order to prepare the dialogue manager for a future clinical trial, we see three immediate resources to help improve the quality of the system (in conjunction with the long-term plans discussed in 5) to be jointly explored:

- Increasing the coverage of transduction trees using data acquired from the pilot.
- Improving fallback methods for failures to extract gist-clauses, such as detecting and responding to a broad thematic topics from the user's input.
- Changing the system's reaction strategy in the case of gist-clause failures, such as sometimes giving "generic" responses rather than making a clarification request.

Chapter 5 Research Plan

This chapter outlines the central research goals for my proposal, and provides an approximate research plan for pursuing these goals.

5.1 Research Goals

5.1.1 Overarching Goal: Creating a Unified, General-Purpose Conversational Agent

Before discussing some more concrete goals of my research proposal in the following sections, I first describe the ambitious long-term goal that motivates and glues together my planned research directions.

In its current iteration, Eta is a general-purpose dialogue manager *shell* (to use the terminology introduced by (Galescu et al., 2018)), in that while the system design is generic and highly modular, it requires integration with a number of domain-specific modules and resources, including specialist reasoners, schemas, pattern transduction rules, etc. Eta applications, such as those described in Chapter 4, are designed to be functional within their domain, but lack any sort of cross-domain capabilities outside of the task they were engineered for.

In Chapter 1, I motivated this proposal by mentioning the long-coveted goal of the AI field of creating a "truly general-purpose human-like conversational agent". This would require a system which is not merely able to have conversations about a topic in one domain, have "spatially aware" question-answering sessions in another domain, and so on, but rather is able to demonstrate each of these capabilities simultaneously.

While this goal is a highly ambitious one, and almost certainly not feasible to accomplish within the scope of my dissertation (it would require a solution to the "knowledge acquisition bottleneck", for instance), a more realistic goal is to demonstrate that the particular research directions described in the following sections can come together to show a clear path forward for achieving the overarching goal. I plan to do this by developing modular, domain-independent methods for achieving the following goals wherever possible, while also continuing to expand the capabilities of the virtual conversational avatars created using Eta.

5.1.2 Exploring Less Brittle Methods for Interpretation & Generation

In the experiments with the SOPHIE system described in Chapter 4, we found that the relatively shallow gist-clause interpretation via transduction trees, while allowing high coverage for relatively minimal cost, were not fully sufficient for handling the linguistic diversity of doctor-patient conversations, and were too brittle with respect to detecting different phrasings of the same underlying gist-clause. Yet, in cases where the system was able to extract a gist-clause, it was typically able to generate an appropriate response, suggesting that the gist-clause interpretation stage is a major bottleneck in the overall performance of the system. Despite this, we believe that the multi-stage interpretive process is valuable for allowing easy configuration of the system and greatly simplifying the semantic parsing process.

Furthermore, while the domain-specific ULF parser used in the blocks world experiments was highly accurate, ensuring wider coverage in more general domains – such as a "room world" with everyday objects and more complex question possibilities – would be a non-trivial task. A generic parser will also allow the benefits of full UFL parsing (viz. logical inference) to be utilized by the SOPHIE agent.

Both of these observations point to the need for more robust methods to be incorporated at both levels of semantic interpretation. I describe some possible directions below, though other approaches may reveal themselves in the future.

Gist-Constrained Decoding

In some simplified sense, the gist-clause interpretation process can be viewed as a retrieval task in practice: an avatar has a fixed number of predefined gist-clauses that it selects by following the appropriate path in a pattern transduction tree. As a first attack on making gist-clause interpretation more robust, one might consider using a statistical retrieval model – where gist-clauses are ranked according to some similarity score to the user utterance conditioned on context – as a fallback. However, an obstacle to this approach is that gist-clauses often "borrow" from content in the user's utterance, such as the final gist-clause obtained in Figure 3.3.

An alternative is to view gist-clause interpretation as a *constrained paraphrase* task. This approach is inspired by Shin et al. (2021), who follow a similar two-stage interpretation where a canonical utterance is first derived from the input using a language model, before mapping the canonical utterance to a semantic form using a symbolic grammar. To produce the canonical utterance, they perform beam-search decoding while constraining the next token in the search to conform to a predefined grammar.

So far, I have conducted some preliminary experiments with *gist-constrained decoding* using the T0 encoder-decoder language model (Sanh et al., 2021). The model is prompted to paraphrase an utterance given a context sentence, and at each generation step, the next token is constrained by a list of predefined gist-clause templates. Specifically, given a partially decoded sequence and a possible next token, the custom decoder checks whether the concatenation of the sequence and token matches a prefix of some gist-clause, where placeholders for "borrowed" content in the gistclause are matched to any sequence of words of corresponding length. If not, the score for that token is set to -1, preventing it from being chosen.

When running this model on a single dialogue (17 turns) from the SOPHIE pilot data, it was able to decode a correct gist-clause in about 30% of the turns – the model was easily fooled by superficial language (e.g., extracting "*I'm sorry you have cancer*." if the utterance uses the word "sorry" at all). However, if used only as a fallback mechanism for when the pattern transduction trees fail to extract a gist-clause, even a 30% increase in gist-clause accuracy could lead to substantial performance gains. It's possible that this accuracy can be further increased by fine-tuning a language model on the VOICE dataset as described in Section 4.3.4.

A critical issue that arose, however, is that the constrained decoding process is intractable given the computational resources that I have access to. In the worst case, generation took up to 30 minutes to complete when running on a single NVIDIA GeForce GTX 1080 GPU (in contrast, generation took about a minute without constrained decoding). In the future, I plan to further investigate the tractability of this approach in order to determine whether or not it's a fruitful direction.

Embedding-based Pattern-Matching

A more conservative approach to making gist-clause extraction more robust may be to incorporate distributional information into the pattern-matching process itself. This would entail first mapping inputs and patterns into some embedding space (using a contextualized embedding), and then computing a "soft match" using a similarity metric in that space, such as cosine similarity. The pattern transduction process could follow one of two strategies, depending on tractability considerations. First, each gist-clause could be weighted by the average similarity score along the path to reach that gist-clause¹, with the system choosing the gist-clause with the highest weight. If this approach is not tractable, however, the system could use the same tree traversal algorithm that it currently uses, but applying a threshold on the computed similarity scores to determine whether to traverse a child node.

Incorporating a Neural ULF Parser

Previous work on statistical semantic parsing demonstrates that neural networks can be used to generate structured representations such as semantic meaning representations. Platanios et al. (2021) and Andreas et al. (2020) show that Transformerbased encoder-decoder networks can be used to predict structured programs for taskoriented dialogue, while Gildea et al. (2018) parse AMR (a graph-based meaning representation similar to ULF) from English using neural cache transition parsers.

My lab has carried out similar efforts in developing a neural cache transition parser for mapping English to ULF; Kim et al. (2021) show that this model achieves results comparable to other state-of-the-art neural semantic parsers. While the parser likely requires further development (and possibly fine-tuning) before it becomes viable in domains such as SOPHIE or the blocks world, it provides an option for broad-coverage semantic parsing for any out-of-domain inputs. This would most likely be implemented using a dual parsing pipeline, similar to the one described in (Valenti et al., 2020), where the general neural ULF parser is used as a fallback if the system's domain-specific hand-engineered ULF parsers are unsuccessful.

Improving Response Quality

Although not a priority (experiments with the system so far indicate that response generation is fairly reliable and realistic, so long as the user is interpreted correctly), I plan to experiment with using language models to improve response generation as well. Due to lack of creativity or oversight in designing transduction trees for response generation in domains such as SOPHIE, the system can sometimes be repetitive or "robotic" in its responses. One way to improve response generation would be to use transduction trees only to select a gist-clause for reaction, and then prompting a language model such as DialoGPT (possibly fine-tuned on a domain-specific dataset) to paraphrase the gist-clause in the context of the user's utterance. I plan to experiment with this in the SOPHIE domain, using the VOICE dataset for fine-tuning, prior to further user studies.

¹An average is used rather than multiplicative product to avoid path length bias.
5.1.3 Leveraging Bottom-Up Inference From Stored Facts

Apart from more robust interpretation and generation mechanisms, the most critical direction of future improvement to the Eta dialogue manager is allowing it to leverage its fact store – i.e., its context, episodic memory, and general knowledge base – for inference in a bottom-up manner. As discussed in Section 4.4, the system needs to be able to draw on its knowledge of the user or previous dialogue events to be able to ensure global coherence and avoid repetition or contradiction.

Currently, the transduction tree mechanism is used to enable inference in a limited capacity, enabled by an augmented pattern-transduction language that can perform tree-to-tree transduction (given that ULF is represented as a tree structure). In the SOPHIE domain, this is used to derive facts about the system's knowledge from interpreted gist-clauses; for example, if the user says that SOPHIE should not seek Chemotherapy, Eta will infer that SOPHIE knows the answer to whether she should seek Chemotherapy – this prevents SOPHIE from later activating a schema for asking about Chemotherapy. However, in contrast to this "hard-coded" approach driven by system inputs, a bottom-up inference mechanism would constantly attempt to derive inferences from known facts. To make such a mechanism scalable, Eta would need both an algorithm for *retrieving* some subset of potentially relevant facts, as well as a deduction system that attempts to generate new facts (e.g., through forward chaining and unification).

Relevant Fact Retrieval

Two considerations may help with efficient retrieval of relevant facts. First, we assume that facts that were recently added to the system's fact store are, all else equal, more fertile grounds for interesting inferences than "stale" facts. In practice, this assumption may be realized by indexing facts by recency values that are periodically updated and using a fixed cutoff for retrieval, or alternatively by maintaining a fixed-size queue of facts that are available for inference, where the oldest facts are pushed out as newer facts are added.

Second, we assume that facts which are deemed to be *salient* to newly added facts may be relevant for inference, regardless of recency – for example, if the SO-PHIE avatar interprets the user as saying that treatment cannot cure her cancer, the fact that chemotherapy is a cancer treatment may be salient for inferring that chemotherapy cannot cure her cancer (and thus, SOPHIE may avoid asking a redundant question such as whether chemotherapy is an option). One possible method of selecting salient facts may be to index facts by predicates within them (particularly verb and noun predicates) and use type hierarchy information (e.g., WordNet hypernymy relations) to find facts that are related to a newly added fact. The salient fact in the above example could be retrieved by querying for facts that are indexed by the predicate "treatment", its hypernyms, and hyponyms (within a certain tree distance).

Forward Inference

Once a set of relevant facts are obtained, some method of deduction from EL formulas is needed to infer new facts. A general method for EL inference is already enabled by the EPILOG system (Schubert and Hwang, 2000), which uses a Rule Instantiation inference rule to generalize forward chaining (while another rule, Goal Chaining, is used for generalized goal-directed backwards chaining). I plan to explore the use of the EPILOG system as a subroutine in the Eta system for generating new inferences in a bottom-up manner.

5.1.4 Utilizing Intensional Models for Schema Selection

The most common conditions of dialogue schemas involve predications about the *beliefs, desires, and intents* of agents within the dialogue. For example, the concept-tutoring schema shown in Figure 3.1 has as a goal condition that the person teaching the concept wants the pupil to understand a concept. A dialogue schema in the SOPHIE domain for inquiring about side-effects chemotherapy, for instance, may suppose that the doctor believes that chemotherapy is an option, or intends for the patient to get chemotherapy. Thus, it's critical for the system to maintain facts about the user's beliefs, desires, and intents – so called "intensional properties", due to their truth-conditional dependence on "private states" of an agent's cognition – in order to select relevant schemas.

However, as mentioned in Chapters 1 and 2, deriving knowledge about these states from user utterances is not often an easy task. Rather than being a straightforward matter of semantic interpretation, these intensional properties are often related to multifaceted lexically-triggered entailments or presuppositions from verbal predicates, depending on which lexical properties are "backgrounded" or "foregrounded" by that predicate (Anand and Hacquard, 2014). The silver lining of these findings is that there appear to be patterns among clusters of related predicates with respect to the types of inferences that they trigger. A model of the intensional properties associated with various predicate clusters could therefore be used to generate belief, desire, and intent inferences from the user's input. I describe ongoing work to create such a model, and some ideas for how this model might be used by the Eta system.

Modelling Lexically-Triggered Inferences from Crowdsourced Annotations

To create such a model, it was first necessary to collect data on lexically-triggered inferences, as assessed by natural English speakers, at a lexicon-wide scale. To do this, we use Amazon Mechanical Turk to crowdsource annotations of belief and desire inferences using a "semantically bleached" templatic format – for instance, a worker might be given the prompt "A lied to B that C happened. How likely is it that A believed that C happened?" and asked to provide a rating on a sliding scale. In (Kane et al., 2021), we collected annotations for 725 unique predicates across 12 finite syntactic frames (varying embedded tense as well as negation), resulting in the MegaIntensionality dataset².

We then modelled inference patterns by finding a soft clustering of predicateframe pairs based on the inference patterns associated with them – i.e., the annotations for belief/desire as well as other types of lexically-triggered inferences such as veridicality and negation-raising. We chose the minimal number of clusters necessary to reliably explain the syntactic distribution of predicates, resulting in 15 predicate clusters. Details on the model are omitted from this proposal for brevity, but are described in (Kane et al., 2021). The inference patterns that we obtained for belief and desire inferences are shown in Figure 5.1. The belief and desire inferences (under different negation and subject configurations) are shown on the y-axis, and verb clusters on the x-axis. We observe, for instance, that verbs in the class of "representationals" (including verbs such as 'thought' and 'suspected') trigger belief inferences that are targeted by negation, while the various positive and negative emotive clusters (including verbs such as "love" and "hate", respectively) trigger desire inferences that are targeted by negation, and belief presuppositions that aren't targeted by negation.

Efforts are currently ongoing to extend the MegaIntensionality dataset with annotations for non-finite frames, while also asking workers to rate intent in addition to belief and desire (for instance, "A commanded B to do C. How likely is it that A intended for B to do C?").

Using Models for Schema Selection

Once a wide enough coverage is achieved with the model described above, I plan to use the models to augment Eta's inference process, and ultimately aid in schema selection. Given a user input that has been parsed into ULF, we can use a clusterbased model of inference patterns to generate additional ULF inferences (pertaining to belief, desire, and intent) from particular verbal predicates within the ULF, depending on which cluster the verb and frame fall into; these generated inferences would subsequently be added to the dialogue context.

²http://megaattitude.io/projects/mega-intensionality/



Figure 5.1: Prototypical belief and desire inference patterns for each verb cluster.

These inferences can also be used as a downstream cue for schema selection or schema matching (the objective described in Section 5.1.6), in which case they provide a suitable abstraction for matching observations to schema conditions, in a similar sense to how word ontologies were used by Lawley et al. (2019) as a useful abstraction in matching event schemas to stories.

5.1.5 Tense-trees and Timegraphs for Temporal Reasoning

Some limitations in Eta's current temporal specialist subsystem were discussed in Section 4.2 in the context of the blocks world QA task. Outside of this particular context, and in more general settings, the system will need to be able to relate events temporally for certain inferences. As we saw in Section 3.1, dialogue schemas may also contain episodes that occur non-linearly and which are related through temporal constraints specified in the schema, in which case relating episodes temporally becomes important for schema matching.

I plan to approach this problem through the use of a tense-tree algorithm for *deindexing* ULF interpretations – that is, extrapolating explicit temporal relations and modifiers from "underspecified" tense operators and temporal adverbs – in conjunction with a timegraph – a directed acyclic graph (DAG) representing episode

ordering relations. In some sense, this is a natural generalization of the algorithm in (Kane et al., 2020): temporal operators and modifiers are processed recursively rather than extracted as a set of constraints, and temporal reasoning becomes a graph breadth-first search (BFS) rather than backtracking on a linear chain.

Tense-tree Deindexing

So far, I've created an implementation of the tense-tree algorithm described in (Hwang and Schubert, 1994) that takes a ULF formula as input and produces a tense-tree structure along with a deindexed EL formula containing explicit temporal relations between episodes³. This algorithm works by applying rules for temporal operators and adverbial modifiers that jointly build up a tense-tree and an EL formula. The nodes in a tense-tree can have up to three children corresponding to past, perfect, and future tenses, as well as embedding links. An example rule for the "past" tense operator is shown below, where predicates or variables subscripted with T indicate indexicals that obtain values depending on the current tense-tree, and the arrow/circle operators indicate operations on the tense-tree (such as visiting another node or inserting an episode at a current node).

$$(\text{past } \Phi)_{T} \iff (\exists e_{T} : [[e_{T} \text{ bef}_{T} \text{ Emb}_{T}] \land [\texttt{Last}_{\checkmark T} \text{ orients.p } e_{T}]] [\Phi_{\bigcirc \checkmark T} ** e_{T}])$$

Tree transformation: (past Φ) $\cdot T = \uparrow (\Phi \cdot (\bigcirc \checkmark T))$

A step-by-step example of the deindexing algorithm given a ULF input is shown in Figure 5.2, beginning with an existing tense-tree from a previous parse. At each step, the ULF operator corresponding to the rule being applied is highlighted in green in the input, and changes to the tense-tree and EL output resulting from that rule are highlighted in red.

Some temporal relations in this example are still "context-charged" even after deindexing. For example, "orients" may yield either 'before', 'during', or 'after' depending on lexical aspect and world knowledge. The 'impinges-on' and 'at-or-before' relations can also be further resolved depending on lexical aspect: both relations mean 'before' in the case of a telically characterized episode (i.e., having an inherent endpoint), but in the case of an atelic episode, the former means 'meets' and the latter means 'same-time'. One way to resolve such cases may be to use a fine-grained statistical model of temporal relations to classify event orderings in the case of 'orients' (first episodes would need to be projected back to their characterizing English

 $^{^{3}}$ Technically the resulting EL formulas are only partially deindexed, as non-temporal indexicals are not affected by this process.



Figure 5.2: A step-by-step example of the tense-tree deindexing algorithm, starting from a ULF interpretation of "John had told Mary that he would call her".



Figure 5.3: An example of a timegraph after insertion of episodes from two sentences. Arrows represent 'before' relations between start and end points of episodes.

sentences), and to use embedding-aided aspectual classifications in the cases of the other two relations.

Temporal Adverbials

One caveat in the deindexing process is the matter of temporal adverbials. Although the tense-tree rules for deindexing temporal and frequency adverbials in (Hwang and Schubert, 1994) are fairly straightforward, some general "normalization" rules are required to map different adverbials from standard ULF to expanded forms that reflect their full predicative meanings. For example, in the case of a frequency adverb – "John saw the movie twice" – a standard ULF interpretation would be (John ((past see.v) (the.d movie.n) twice.adv-f)). The adverb twice.adv-f would need to be expanded to (adv-f (two.a (plur episode.n))) in order to obtain the desired predication (e1 (two.a (plur episode.n))) in the deindexed EL formula (where e1 is characterized by multiple "John sleeping" episodes).

While straightforward in many cases, habituals involving quantificational adverbs (qadverbs) such as "always" or "typically" present a particular challenge. It has been suggested that qadverbs quantify over situations that are restricted by presuppositions of the scope and context of the qadverb (Ahn, 2001) – for an example from the blocks world domain, the statement "I frequently moved the Twitter block" can be interpreted as "When I moved a block, I frequently moved the Twitter block", where the presupposition (that I moved a block) is borrowed by the quantificational restrictor. Thus, to handle such cases I will likely need a method for generating

presuppositions. I plan to look into using existing ULF presupposition models such as (Kim et al., 2019), though it's not yet clear how effective this approach will be.

Timegraph

A timegraph is a DAG that allows for efficient computation of temporal relations between episodes (Miller and Schubert, 1990); the nodes of a timegraph represent episodes with associated beginning and ending time points, while edges represent before/after/equivalency relations between time points.

An instance of a timegraph is shown in Figure 5.3, using the same example that was deindexed in Figure 5.2 (along with a preceding sentence). The directed edges represent 'before' relations between episodes, corresponding to the ones obtained in the deindexing process.

Although the timegraph data structure is well-described, a full implementation still needs to be developed. Subsequently, I plan to incorporate a timegraph into the Eta dialogue manager. After the tense-tree algorithm is used to deindex ULF interpretations, the resulting episodes and their temporal relations are inserted into the timegraph. I then plan to utilize the timegraph in temporal relations between episodes when performing inference or attempting to match schema episodes.

5.1.6 Dynamic Schema Matching for Handling Unexpected Events

While the previous goals may suffice for improving the system's performance in more-or-less anticipated trajectories of conversation, a method is still needed for adapting to completely unexpected events – i.e., those outside of the scope of the system's current schema. For instance, in the SOPHIE conversation it's possible that a user may unexpectedly deviate from the topic of conversation and bring up a discursive topic. Although default response options are always available to bring the conversation back on track, a "truly intelligent" system may engage with the user on this new topic before attempting to return to the main conversation, requiring activation of a new, unrelated dialogue schema.

Therefore, as a more distant goal – but ultimately critical for fully realizing the adaptability that the schema representation provides – I plan to develop an algorithm for selecting and activating new dialogue schemas when appropriate. I envision this as a two step process. First, the system should keep a limited pool of candidate schemas that it deems potential matches to the events being observed. To make this tractable, this likely will need to be limited to "trigger conditions" of the schema –

a subset of schema conditions that, when observed to be true, provide a cue for that schema to be added to the list of candidate schemas. A candidate schema may be evicted from this pool once a trigger condition ceases to be true, or possibly when the pool reaches an upper bound on size (in which case schemas would be evicted in a first-in-first-out order).

Second, a candidate schema may be *activated* if the schema is a sufficiently strong match to observations – where matching is envisioned to follow a similar method to the one employed in (Lawley et al., 2019). A successful match would result in either modification or replacement of the current plan. However, since a schema activated in this manner may potentially start *in medias res* (the events observed by the system, for instance, might match the first several expected episodes in the schema), this would first require an alignment of the dialogue context to a particular episode of the schema, so that the planner can detect where to proceed. One possibility that I plan to investigate further is the use of a Transformer-based schema attention model, based on (Mehri and Eskénazi, 2021), that uses attention between dialogue context and a schema graph (i.e., a graph representing the order of episodes within the schema) to determine an alignment.

5.2 Research Timeline

The research goals in Section 5.1 are ordered in terms of their overall priority with respect to improving the quality of the avatars built using Eta. Thus, my timeline for working on these goals roughly falls in the same order, although is not completely sequential.

5.2.1 Fall 2022:

During this semester, I will be principally focused on preparing the SOPHIE avatar for a clinical trial that we expect to carry out in Spring 2023 (in the most optimistic scenario) or Fall 2023.

My primary focus during this semester will be the goal described in Section 5.1.2 – I will begin by continuing experiments with gist-constrained decoding; specifically, investigating more tractable ways to generate gist-clauses to see whether this approach can be used in practice. If not, I aim to run experiments with the embedding-based pattern-matching approach as an alternative. These approaches to gist-clause extraction will be evaluated using the transcript data collected from the SOPHIE pilot study.

I also plan to experiment with using a fine-tuned language model to paraphrase responses to see if it improves the quality of responses – although it's likely not the most critical path forward, it should be a relatively quick experiment given that I've already fine-tuned DialoGPT on the VOICE dataset.

By this point, we should also have an updated MegaIntensionality model containing nonfinite frames collected from the currently ongoing experiment, which will allow for further progress on Section 5.1.4. A natural starting point would be carrying out a small-scale evaluation using the clusters of inference patterns to generate inferences from simple ULF expressions from a corpus, in a similar manner to (Kim et al., 2019). A final, similar objective for this semester would be conducting a small scale corpus-based evaluation on the temporal adverbial deindexing mechanism described in Section 5.1.5. Ideally, these evaluations will help assess whether these approaches are viable or not, and if so, provide a starting point for incorporating them into the Eta system.

5.2.2 Spring 2023:

In Spring 2023, my focus will remain preparing the SOPHIE avatar for a clinical trial – I will assist with the experiment design, which will resemble the pilot experiment at a larger scale. By this stage, I hope to have made the gist-clause interpretation process fairly robust. My focus will shift towards improving the second stage of semantic interpretation (ULF parsing) with a general-purpose semantic parser. In the ideal scenario, the neural ULF parser proposed in (Kim et al., 2021) is improved through architectural improvements and additional data. However, a preliminary Treebank-based compositional ULF parser developed by Len Schubert⁴ also provides a possible symbolic alternative.

With general ULF parsing enabled in domains such as SOPHIE, the path forward on my second goal – using bottom-up inference to assist with dialogue planning – opens up. As described in Section 5.1.3, I will first experiment with different retrieval methods for inference (perhaps evaluating performance using a "knowledge base" of a large amount of "facts", either procedurally generated or drawn from an existing resource), and then look into adapting the EPILOG system for use by Eta.

5.2.3 Fall 2023:

In Fall 2023, I will analyze and evaluate the data from the clinical trial – which will have already been carried out, or will be carried out early this semester.

The direction of my research at this point will largely depend on the results of this trial. Assuming that the incorporation of my work in the previous semesters substantially improves the performance of the system, which I expect is attainable

⁴https://github.com/genelkim/lenulf

given the conjunction of my first and second goals, I will focus on some of my more specialized research objectives, such as the temporal reasoning work and incorporation of intensional models. After incorporating them into Eta, I plan to test them with some sort of dialogue-based evaluation: for the former, this would require some sort of generalization of the historical QA in a more realistic domain where a wider range of temporal adverbials and event relations are possible. The latter may be difficult to test in isolation, but some small-scale studies in the SOPHIE domain may be possible where the comparative dialogue quality before/after incorporating intensional models is evaluated.

5.2.4 Spring 2024 and Beyond:

Beyond Spring 2024, my final research goal – improving candidate schema selection, schema activation, and schema alignment – becomes a focus assuming satisfactory results have been obtained on the other research goals; otherwise it may be dropped from my thesis so as to prioritize the other goals. I will begin by developing an algorithm for monitoring candidate schemas based on their trigger conditions, as described in 5.1.6, and working to incorporate a schema matching algorithm similar to (or perhaps equivalent to) the one developed by Lawley et al. (2019). A possible method of evaluation for this research goal is creating an "adversarial" dialogue setting where conversation participants attempt to disrupt the course of conversation with an unrelated topic or request – however, this would need to occur in a fairly topically restricted domain, as the problem of actually acquiring a large amount of schemas across various topics lies outside of the scope of my proposal.

Chapter 6 Conclusion

In this proposal, I discussed the strengths and limitations of previous approaches to dialogue management, and then presented an approach that uses *dialogue schemas* – a generalization of hierarchical plans that incorporates matching of expectations – to guide dialogue. I described the overall architecture and components of the resulting dialogue manager – Eta – and then introduced virtual conversational avatars that were developed using Eta: a "blocks world" collaborative agent and a standardized virtual patient for teaching doctor communication skills. Some promising initial results in each of these domains were shown, along with areas that require future improvement. I discuss some specific directions for future work aimed at improving the overall capabilities of the Eta dialogue manager, and outlined a plan for achieving these goals.

Bibliography

- [1] Daniel Adiwardana et al. Towards a Human-like Open-Domain Chatbot. 2020. arXiv: 2001.09977 [cs.CL].
- [2] David Ahn. "Computing adverbial quantifier domains." In: *Proceedings of the* Sixth ESSLLI Student Session. 2001, pp. 1–12.
- [3] 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.
- [4] 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. URL: https://www.sciencedirect. com/science/article/pii/0004370284900080.
- [5] 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.
- [6] James F. Allen, G. Ferguson, and Amanda Stent. "An architecture for more realistic conversational systems." In: *IUI '01*. 2001.
- 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. URL: https://www. sciencedirect.com/science/article/pii/0004370280900429.
- [8] Pranav Anand and Valentine Hacquard. "Factivity, belief and discourse." In: *The Art and Craft of Semantics: A Festschrift for Irene Heim.* Ed. by Luka Crnič and Uli Sauerland. Vol. 1. Cambridge, MA: MIT Working Papers in Linguistics, 2014, pp. 69–90.
- Jacob Andreas et al. "Task-Oriented Dialogue as Dataflow Synthesis." In: *Transactions of the Association for Computational Linguistics* 8 (Sept. 2020), pp. 556–571.

- [10] John Langshaw Austin. *How to do things with words*. William James Lectures. Oxford University Press, 1962.
- [11] Frederic C. Bartlett and Walter Kintsch. Remembering: A Study in Experimental and Social Psychology. 2nd ed. Cambridge University Press, 1995. DOI: 10.1017/CB09780511759185.
- 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.
- [13] D. Bohus and Alexander I. Rudnicky. "The RavenClaw dialog management framework: Architecture and systems." In: *Comput. Speech Lang.* 23 (2009), pp. 332–361.
- [14] Julia Campbell et al. "Developing INOTS to support interpersonal skills practice." In: Apr. 2011, pp. 1–14. DOI: 10.1109/AER0.2011.5747535.
- Stephanie Carnell et al. "Adapting Virtual Patient Interviews for Interviewing Skills Training of Novice Healthcare Students." In: vol. 9238. Aug. 2015. DOI: 10.1007/978-3-319-21996-7_5.
- [16] Herbert H. Clark. Using Language. 'Using' Linguistic Books. Cambridge University Press, 1996. DOI: 10.1017/CB09780511620539.
- [17] 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.
- [18] 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. URL: https: //www.sciencedirect.com/science/article/pii/S0364021379800063.
- [19] G. Ferguson and James F. Allen. "TRIPS: An Integrated Intelligent Problem-Solving Assistant." In: AAAI/IAAI. 1998.
- [20] 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. URL: https://www.sciencedirect.com/science/ article/pii/0004370271900105.

- [21] 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.
- [22] 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.
- [23] Daniel Gildea, Giorgio Satta, and Xiaochang Peng. "Cache Transition Systems for Graph Parsing." In: *Computational Linguistics* 44.1 (Apr. 2018), pp. 85–118.
 DOI: 10.1162/COLI_a_00308. URL: https://aclanthology.org/J18-1004.
- [24] B. Grosz and C. Sidner. "Plans for Discourse." In: 1990.
- 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. URL: https://www.sciencedirect.com/science/article/pii/0004370295001034.
- [26] 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.
- [27] Vrindavan Harrison et al. Athena: Constructing Dialogues Dynamically with Discourse Constraints. 2020. arXiv: 2011.10683 [cs.CL].
- [28] Michael Hoerger et al. "Values and options in cancer care (VOICE): study design and rationale for a patient-centered communication and decision-making intervention for physicians, patients with advanced cancer, and their caregivers." In: *BMC cancer* 13.1 (2013), pp. 1–14.
- [29] Chung Hee Hwang and Lenhart K. Schubert. "Interpreting Tense, Aspect and Time Adverbials: A Compositional, Unified Approach." In: *Proceedings of the First International Conference on Temporal Logic.* ICTL '94. Berlin, Heidelberg: Springer-Verlag, 1994, pp. 238–264. ISBN: 354058241X.
- [30] 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 Guenthner. Dordrecht: Springer Netherlands, 2011, pp. 125– 394. ISBN: 978-94-007-0485-5. DOI: 10.1007/978-94-007-0485-5_3.

- [31] Benjamin Kane, William Gantt, and Aaron S. White. "Intensional Gaps: Relating veridicality, factivity, doxasticity, bouleticity, and neg-raising." In: Semantics and Linguistic Theory 31 (2021). URL: https://par.nsf.gov/biblio/ 10300471.
- Benjamin Kane, Georgiy Platonov, and Lenhart K. Schubert. *History-Aware Question Answering in a Blocks World Dialogue System.* 2020. arXiv: 2005.
 12501 [cs.AI].
- [33] Gene Kim et al. "A Transition-based Parser for Unscoped Episodic Logical Forms." In: Proceedings of the 14th International Conference on Computational Semantics (IWCS). Groningen, The Netherlands (online): Association for Computational Linguistics, June 2021, pp. 184–201. URL: https://aclanthology. org/2021.iwcs-1.18.
- [34] Gene Kim et al. "Generating Discourse Inferences from Unscoped Episodic Logical Formulas." In: Proceedings of the First International Workshop on Designing Meaning Representations. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 56–65. DOI: 10.18653/v1/W19-3306. URL: https://aclanthology.org/W19-3306.
- [35] Staffan Larsson et al. "GoDiS An Accommodating Dialogue System." In: (Dec. 2002). DOI: 10.3115/1117562.1117564.
- [36] 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.
- [37] 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.
- [38] Anton Leuski and David Traum. "NPCEditor: Creating Virtual Human Dialogue Using Information Retrieval Techniques." In: AI Magazine 32.2 (Mar. 2011), pp. 42–56. DOI: 10.1609/aimag.v32i2.2347.
- [39] 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.
- [40] Kellen Maicher et al. "Developing a Conversational Virtual Standardized Patient to Enable Students to Practice History-Taking Skills." In: Simulation in healthcare: journal of the Society for Simulation in Healthcare 12 (Apr. 2017), pp. 124–131. DOI: 10.1097/SIH.00000000000195.

- [41] Colin Matheson, Massimo Poesio, and David Traum. "Modelling Grounding and Discourse Obligations Using Update Rules." In: (May 2002).
- [42] M. McTear. "Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit." In: *ICSLP*. 1998.
- [43] Shikib Mehri and Maxine Eskénazi. "Schema-Guided Paradigm for Zero-Shot Dialog." In: SIGdial. 2021.
- [44] Stephanie A. Miller and Lenhart K. Schubert. "Time revisited." In: Computational Intelligence 6 (1990), pp. 108–118.
- [45] Marvin Minsky. A Framework for Representing Knowledge. Tech. rep. Cambridge, MA, USA, 1974.
- [46] Fabrizio Morbini et al. "FLoReS: A Forward Looking, Reward Seeking, Dialogue Manager." In: (Aug. 2014). DOI: 10.1007/978-1-4614-8280-2_28.
- [47] Reinhard Muskens. "Tense and the Logic of Change." In: Lexical Knowledge in the Organization of Language. Ed. by Urs Egli et al. Amsterdam/Philadelphia: John Benjamins, 1995, pp. 147–183.
- [48] 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.
- [49] 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.
- [50] Emmanouil Antonios Platanios et al. "Value-Agnostic Conversational Semantic Parsing." In: ACL-IJCNLP 2021. Aug. 2021.
- [51] Georgiy Platonov, Benjamin Kane, and Lenhart Schubert. "Generating Justifications in a Spatial Question-Answering Dialogue System for a Blocks World." In: Proceedings of the Reasoning and Interaction Conference (ReInAct 2021). Gothenburg, Sweden: Association for Computational Linguistics, Oct. 2021, pp. 53-57. URL: https://aclanthology.org/2021.reinact-1.8.
- [52] Georgiy Platonov and Lenhart Schubert. "Computational Models for Spatial Prepositions." In: Proceedings of the First International Workshop on Spatial Language Understanding. 2018, pp. 21–30.

- [53] 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.
- [54] Georgiy Platonov et al. "Modeling Semantics and Pragmatics of Spatial Prepositions via Hierarchical Common-Sense Primitives." In: Proceedings of Second International Combined Workshop on Spatial Language Understanding and Grounded Communication for Robotics. Online: Association for Computational Linguistics, Aug. 2021, pp. 32–41. DOI: 10.18653/v1/2021.splurobonlp-1.4. URL: https://aclanthology.org/2021.splurobonlp-1.4.
- [55] Massimo Poesio and David Traum. "Towards an Axiomatization of Dialogue Acts." In: (May 1998).
- [56] 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.
- [57] Martha E. Pollack. "Plans As Complex Mental Attitudes." In: Intentions in Communication. MIT Press, 1990, pp. 77–103.
- [58] 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.
- [59] S. Z. Razavi et al. "Dialogue Design and Management for Multi-Session Casual Conversation with Older Adults." In: *ArXiv* abs/1901.06620 (2019).
- [60] S. Z. Razavi et al. "Managing Casual Spoken Dialogue Using Flexible Schemas , Pattern Transduction Trees , and Gist Clauses." In: 2017.
- [61] 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.
- [62] Ronald Rensink. "Scene Perception." In: Jan. 2001, pp. 151–155.
- [63] 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.

- [64] 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.
- [65] Stephen Roller et al. *Recipes for building an open-domain chatbot.* 2020. arXiv: 2004.13637 [cs.CL].
- [66] Brent Rossen, David Lind, and Benjamin Lok. "Human-Centered Distributed Conversational Modeling: Efficient Modeling of Robust Virtual Human Conversations." In: Sept. 2009, pp. 474–481. ISBN: 978-3-642-04379-6. DOI: 10. 1007/978-3-642-04380-2_52.
- [67] Victor Sanh et al. "Multitask Prompted Training Enables Zero-Shot Task Generalization." In: CoRR abs/2110.08207 (2021). arXiv: 2110.08207. URL: https://arxiv.org/abs/2110.08207.
- [68] 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.
- [69] 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=YZ99AAAMAAJ.
- [70] 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.
- [71] John R. Searle. Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, 1969. DOI: 10.1017/CB09781139173438.
- [72] Richard Shin et al. "Constrained Language Models Yield Few-Shot Semantic Parsers." In: 2021 Empirical Methods in Natural Language Processing. Jan. 2021.
- [73] D. Traum and James F. Allen. "Discourse Obligations in Dialogue Processing." In: ACL. 1994.

- [74] 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. URL: https://doi.org/10.1007/978-94-010-0019-2_15.
- [75] Roy M. Turner. Adaptive Reasoning for Real-World Problems: A Schema-Based Approach. Hillsdale, NJ: Lawrence Erlbaum, 1994.
- [76] 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://semdial.org/ anthology/Z20-Valenti_semdial_0023.pdf.
- [77] Teun Adrianus Van Dijk and Walter Kintsch. *Strategies of discourse comprehension*. Academic press New York, 1983.
- [78] Marilyn Walker and Steve Whittaker. "Mixed Initiative in Dialogue: An Investigation into Discourse Segmentation." In: *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*. ACL '90. Pittsburgh, Pennsylvania: Association for Computational Linguistics, 1990, pp. 70–78. DOI: 10.3115/981823.981833. URL: https://doi.org/10.3115/981823.981833.
- [79] 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.
- [80] 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. URL: https://doi.org/10.1007/978-3-319-57837-8\ 18.
- [81] Terry Winograd. "Understanding natural language." In: Cognitive psychology 3.1 (1972), pp. 1–191.
- [82] 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.

[83] Yizhe Zhang et al. "DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation." In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. Online: Association for Computational Linguistics, July 2020, pp. 270–278. DOI: 10.18653/v1/2020.acl-demos.30. URL: https://aclanthology.org/2020.acl-demos.30.