Planning Dialog Actions

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh EH8 9LW, Scotland, UK
steedman@inf.ed.ac.uk

Ronald P. A. Petrick
School of Informatics
University of Edinburgh
Edinburgh EH8 9LW, Scotland, UK
rpetrick@inf.ed.ac.uk

Abstract

The problem of planning dialog moves can be viewed as an instance of the more general AI problem of planning with incomplete information and sensing. Sensing actions complicate the planning process since such actions engender potentially infinite state spaces. We adapt the Linear Dynamic Event Calculus (LDEC) to the representation of dialog acts using insights from the PKS planner, and show how this formalism can be applied to the problem of planning mixed-initiative collaborative discourse.

1 Introduction

Successful planning in dynamic domains often requires reasoning about sensing acts which, when executed, update the planner's knowledge state without necessarily changing the world state. For instance, reading a piece of paper with a telephone number printed on it may provide the reader with the prerequisite information needed to successfully complete a phone call. Such actions typically have very large, even infinite, sets of possible outcomes in terms of the actual sensed value, and threaten to make search impracticable. There have been several suggestions in the AI literature for how to handle this problem, including Moore (1985); Morgenstern (1988); Etzioni et al. (1992); Stone (1998); and Petrick & Bacchus (2002; 2004).

Stone (2000) points out that the problem of planning effective conversational moves is also a problem of planning with sensing or knowledge-producing actions, a view that is also implicit in early "beliefs, desires and intentions" (BDI) -based approaches (e.g., Litman & Allen (1987); Bratman, Israel & Pollack (1988); Cohen & Levesque (1990); Grosz & Sidner (1990)). Nevertheless, most work on dialog planning has in practice tended to segregate domain planning and discourse planning, treating the former as an AI black box, and capturing the latter in large state-transition machines mediated or controlled via a blackboard or "information state" representing mutual belief, updated by specialized rules more or less directly embodying some form of speech-act theory, dialog game, or theory of textual coherence (e.g., Lambert & Carberry (1991); Traum & Allen (1992); Green & Carberry (1994); Young & Moore (1994); Chu-Carroll & Carberry (1995); Matheson, Poesio & Traum (2000); Beun (2001)); Asher & Lascarides (2003); Maudet (2004)). Such accounts often lend themselves to optimization using statistical models (e.g., Singh et al. (2002)).

One of the ostensible reasons for making this separation is that *indirect* speech acts, i.e., achieving coherence via implicatures, abound in conversation. (For instance, Green and Carberry cite studies showing around 13% of answers to Yes/No questions are indirect.) Nevertheless, that very same ubiquity of the phenomenon suggests it is a manifestation of the same planning apparatus as the domain planner, and that it should not be necessary to construct a completely separate specialized planner for dialog acts.

This paper addresses the problem of dialog planning by applying techniques developed in the AI planning literature for handling sensing and incomplete information. To this end, we work with planning domains axiomatized in the language of the

Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue, pages 265–272,
Antwerp, September 2007.

Linear Dynamic Event Calculus (LDEC), but extended with constructs inspired by the knowledge-level conditional planner PKS.

2 Linear Dynamic Event Calculus (LDEC)

The Linear Dynamic Event Calculus (LDEC) (Steedman, 1997; Steedman, 2002) is a logical formalism that combines the insights of the Event Calculus of Kowalski & Sergot (1986), itself a descendant of the Situation Calculus (McCarthy and Hayes, 1969), and the STRIPS planner of Fikes & Nilsson (1971), together with the Dynamic and Linear Logics developed by Girard (1987), Harel (1984), and others.

The particular dynamic logic that we work with here exclusively uses the deterministic "necessity" modality $[\alpha]$. For instance, if a program α computes a function f over the integers, then an expression like "$n \geq 0 \Rightarrow [\alpha](y = f(n))$" indicates that "in any situation in which $n \geq 0$, after every execution of α that terminates, $y = f(n)$." We can think of this modality as defining a logic whose models are Kripke diagrams, where accessibility between situations is represented by events defined in terms of the conditions which must hold before an event can occur (e.g., "$n \geq 0$"), and the consequences of the event that hold as a result (e.g., "$y = f(n)$").

Thus, *actions* (or *events*) in LDEC provide the sole means of change and affect the *fluents* (i.e., properties) of the world being modelled. Like other dynamic logics, LDEC does not use explicit situation terms to denote the state-dependent values of fluents, but instead, chains together finite sequences of actions using a *sequence* operator ";". For instance, $[\alpha_1; \alpha_2; \ldots; \alpha_n]$ denotes a sequence of n actions and $[\alpha_1; \alpha_2; \ldots; \alpha_n]\phi$ means that ϕ must necessarily hold after every execution of this sequence.

One of the novel features of LDEC is that it mixes two types of logical implication. Besides standard (or intuitionistic) implication $\Rightarrow$, LDEC follows Bibel et al. (1989) and others in using *linear* logical implication, denoted by the symbol $\multimap$. Linear implication extends LDEC's representational power and provides a solution to the *frame problem* (McCarthy and Hayes, 1969), as we'll see below.

An LDEC *domain* is formally described by a collection of axioms. For each action α, a domain includes an *action precondition axiom* of the form:

$$L_1 \wedge L_2 \wedge \ldots \wedge L_k \Rightarrow \mathit{affords}(\alpha),$$

where each L_i is a fluent or its negation (we discuss *affords* below), and an *effect axiom* of the form:

$$\{\mathit{affords}(\alpha)\} \wedge \phi \multimap [\alpha]\psi,$$

where ϕ and ψ are conjunctions of fluents or their negations. LDEC domains can also specify a collection of *initial situation axioms* of the form:

$$L_1 \wedge L_2 \wedge \ldots \wedge L_p,$$

where each L_i is a ground fluent literal. Finally, LDEC domains can include a set of background axioms (e.g., for defining the properties of other modal operators), and a set of simple state constraint axioms (e.g., for encoding inter-fluent relationships). We will not discuss the details of these axioms here.

Action precondition axioms specify the applicability conditions of actions using a special *affords* fluent. Effect axioms use linear implication to build certain "update rules" directly into the LDEC representation. In particular, the fluents of ϕ in the antecedent of an effect axiom are treated as consumable resources that are replaced by the fluents of ψ in the consequent when an action α is applied.[1] $\{\mathit{affords}(\alpha)\}$ means that it is not defined whether $\mathit{affords}(\alpha)$ still holds after α. All other fluents are unchanged. Thus, LDEC's use of linear implication builds a STRIPS-style (Fikes and Nilsson, 1971) treatment of action effects into the semantics of the language, which lets us address the frame problem without having to write explicit frame axioms.

Previous work has demonstrated LDEC's versatility as a language for modelling dialog, by introducing notions of speaker/hearer supposition and common ground (Steedman, 2006). This is achieved by defining a new set of modal operators of the form [X], that designate the participants in the dialog and provide a reference point for the shared beliefs that exist between those participants. For instance, [S] and [H] refer to the "speaker" and "hearer", respectively, while $[C_{SH}]$ refers to the common ground between speaker and hearer.[2] Using these modalities

[1] We treat consumed fluents as being made false.

[2] Additional participant modalities can be defined as needed. A set of LDEC background axioms is provided as part of a domain to govern the behaviour of these modalities.

we can write LDEC formulae that capture common propositions that arise in dialog. For instance, [S] p means "the speaker supposes p", [S] [H] p means "the speaker supposes that the hearer supposes p", and [C_{SH}] [X] p means "it is common ground between the speaker and hearer that X supposes p".

In this paper we extend LDEC even further. First, we recognize the need to model *knowledge* in LDEC, which is a necessary prerequisite for planning with sensing actions, including those needed for effective discourse. Second, we require that our extended representation lend itself to tractable reasoning, in order to facilitate a practical implementation. Finally, although LDEC supports classical plan generation through proof (Steedman, 2002), prior work has not addressed the problem of translating LDEC domains into a form that can take advantage of recent planning algorithms for reasoning with incomplete information and sensing. For a solution to these problems we turn to the PKS planner.

3 Planning with Knowledge and Sensing (PKS)

PKS (Planning with Knowledge and Sensing) is a knowledge-level planner that can build conditional plans in the presence of incomplete information and sensing (Petrick and Bacchus, 2002; Petrick and Bacchus, 2004). Unlike traditional approaches that focus on modelling the world state and how actions change that state, PKS works at a much higher level of abstraction: PKS models an agent's knowledge state and how actions affect that knowledge state.

The key idea behind the PKS approach is that the planner's knowledge state is represented using a first-order language. Since reasoning in a general first-order language is impractical, PKS employs a restricted subset of this language and limits the amount of inference it can perform. This approach differs from those approaches that use propositional representations (i.e., without functions and variables) over which complete reasoning is feasible, or works that attempt to represent complete sets of possible worlds (i.e., sets of states compatible with the planner's incomplete knowledge) using BDDs, Graphplan-like structures, clausal representations, or other such techniques.

What makes the PKS approach particularly novel is the level of abstraction at which PKS operates. By reasoning at the knowledge level, PKS can avoid some of the irrelevant distinctions that occur at the world level, which gives rise to efficient inference and plans that are often quite "natural". Although the set of inferences PKS supports is weaker than that of many possible-worlds approaches, PKS can make use of non-propositional features such as functions and variables, allowing it to solve problems that can be difficult for world-level planners.

Like LDEC, PKS is based on a generalization of STRIPS. In STRIPS, the world state is modelled by a single database. In PKS, the planner's knowledge state, rather than the world state, is represented by a set of five databases whose contents have a fixed, formal interpretation in a modal logic of knowledge. To ensure efficient inference, PKS restricts the types of knowledge (especially disjunctions) each database can model. We briefly describe three of these databases (K_f, K_v, and K_w) here.

K_f: This database is like a standard STRIPS database except that both positive and negative facts are stored and the closed world assumption is not applied. K_f can include any ground literal ℓ, where $\ell \in K_f$ means "ℓ is known". K_f can also contain knowledge of function values.

K_v: This database stores information about function values that will become known at execution time, such as the plan-time effects of sensing actions that return numeric values. During planning, PKS can use K_v knowledge of finite-range functions to build multi-way conditional branches into a plan. K_v function terms also act as "run-time variables"—placeholders for function values that will only be available at execution time.

K_w: This database models the plan-time effects of "binary" sensing actions. $\phi \in K_w$ means that at plan time the planner either knows ϕ or knows $\neg\phi$, and that at execution time this disjunction will be resolved. PKS uses such "know-whether" facts to construct binary conditional branches in a plan.

PKS also includes a database (K_x) of known "exclusive-or" disjunctions and a database (*LCW*) for modelling known instances of "local closed world" information (Etzioni et al., 1994).

Actions in PKS are modelled as queries and updates to the databases. *Action preconditions* are specified as a list of *primitive queries* about the state

of the databases: (i) Kp, is p known to be true?, (ii) $K_v t$, is the value of t known?, (iii) $K_w p$, is p known to be true or known to be false (i.e., does the planner know-whether p)?, or (iv) the negation of (i)–(iii). *Action effects* are described by a set of STRIPS-like *database updates* that specify the formulae to be added to and deleted from the databases. These updates capture the changes to the planner's knowledge state that result from executing the action.

Using this representation, PKS constructs plans by applying actions in a simple forward-chaining manner: provided an action's preconditions are satisfied by the planner's knowledge state, an action's effects are applied to form a new knowledge state. Conditional branches can be added to a plan provided the planner has K_w or (particular types of) K_v information. For instance, if the planner has K_w information about a formula p then it can add a binary branch to a plan. Along one branch, p is assumed to be known while along the other branch $\neg p$ is assumed to be known. PKS can also use K_v information to denote certain execution-time quantities in a plan. Planning continues along each branch until all branches satisfy the goal.

4 Planning Speech Acts with LDEC/PKS

Our approach to planning dialog acts aims to introduce certain features of PKS within LDEC, with the goal of generating plans using the PKS framework. In this paper we primarily focus on the representational issues concerning LDEC, and simply sketch our approach for completing the link to PKS.

The most important insight PKS provides is its action representation based on simple *knowledge primitives*: K/K_f "know", K_v "know value", and K_w "know whether". In particular, PKS's tractable treatment of this information—which underlies its databases and queries—is essential to its ability to build plans with incomplete knowledge and sensing.

In order to model similar conditions of incomplete information in LDEC, we introduce a set of PKS-style knowledge primitives into LDEC in the form of *knowledge fluents* (Demolombe and Pozos Parra, 2000). Knowledge fluents are treated as ordinary fluents but are understood to have particular meanings with respect to the knowledge state. For instance, in our earlier example of reading a piece of paper with a telephone number printed on it, we could use a knowledge fluent $KhavePaper$ to indicate that an agent knows it has the required piece of paper, $K_v phoneNumber$ to represent the result of reading the phone number from the paper (i.e., the agent "knows the value of the phone number"), and $K_w connected$ to denote the result of actually dialling the phone number (i.e., the agent "knows whether the call connected successfully").

In a dialog setting, we must also ground all knowledge-level assertions to particular participants in the dialog, or to the common ground. Otherwise, such references will have little meaning in a multi-agent context. Thus, we couple speaker/hearer modalities together with knowledge fluents to write LDEC expressions like $[\mathrm{S}]\,Kp$ — "the speaker knows p", $[\mathrm{H}]\,K_v t$ — "the hearer knows the value of t", or more complex expressions like $[\mathrm{C_{SH}}]\,[\mathrm{H}]\,K_w p$ — "it's common ground between the speaker and hearer that the hearer knows whether p".

Although we treat knowledge fluents as ordinary fluents in LDEC, we retain their knowledge-level meanings with respect to their use in PKS. Thus, knowledge fluents serve a dual purpose in LDEC. First, they act as queries for establishing the truth of particular knowledge-level assertions (e.g., an action precondition axiom like $[\mathrm{X}]\,Kp \Rightarrow \mathit{affords}(\alpha)$ means "if X knows p then this affords action α"). Second, they act as updates that specify how knowledge changes due to action (e.g., an effect axiom like $\{\mathit{affords}(\alpha)\} \multimap [\alpha][X]K_v t$ means "executing α causes X to come to know the value of t"). This correlation between LDEC and PKS is not a coincidence but one, we hope, that will let us use PKS as a target planner for LDEC domains.

We illustrate our LDEC extensions in the following domain axiomatization, which is sufficient to support planning with dialog acts.

4.1 Background Axioms

(1) $[\mathrm{X}]\,p \Rightarrow p$ Supposition Veridicality

(2) $[\mathrm{X}]\,\neg p \Rightarrow \neg\,[\mathrm{X}]\,p$ Supposition Consistency

(3) $\neg\,[\mathrm{X}]\,p \Rightarrow [\mathrm{X}]\,\neg\,[\mathrm{X}]\,p$ Negative Introspection

(4) $[\mathrm{C_{SH}}]\,p \Leftrightarrow ([\mathrm{S}]\,[\mathrm{C_{SH}}]\,p \wedge [\mathrm{H}]\,[\mathrm{C_{SH}}]\,p)$ Common Ground

(5) $[X]\,[C_{XY}]\,p \Rightarrow [X]\,p$

Common Ground Veridicality

4.2 Initial Facts

(6) a. "I suppose Bonnie doesn't know what train I will catch"
b. $[S] \neg [B]\, K_v train$

(7) a. "If I know what time it is, I know what train I will catch."
b. $[S]\, K_v time \Rightarrow [S]\, K_v train$

(8) a. "I don't know what train I will catch."
b. $[S] \neg K_v train$

(9) a. "I suppose you know what time it is."
b. $[S]\,[H]\, K_v time$

(10) a. "I suppose it's not common ground that I don't know what time it is."
b. $[S] \neg [C_{SH}] \neg [S]\, K_v time$

4.3 Rules

(11) a. "If X supposes p, and X supposes p is not common ground, X can tell Y p"
b. $[X]\,p \wedge [X] \neg [C_{XY}]\,p$
$\Rightarrow \mathit{affords}(\mathit{tell}(X, Y, p))$

(12) a. "If X tells Y p, Y stops not knowing it and starts to know it."
b. $\{\mathit{affords}(\mathit{tell}(X, Y, p))\} \wedge \neg [Y]\,p$
$\multimap [\mathit{tell}(X, Y, p)]\,[Y]\,p$

(13) a. "If X doesn't know p and X supposes Y does, X can ask Y about it."
b. $\neg [X]\,p \wedge [X]\,[Y]\,p$
$\Rightarrow \mathit{affords}(\mathit{ask}(X, Y, p))$

(14) a. "If X asks Y about p, it makes it common ground X doesn't know it"
b. $\{\mathit{affords}(\mathit{ask}(X, Y, p))\}$
$\multimap [\mathit{ask}(X, Y, p)]\,[C_{XY}] \neg [X]\,p$

Axioms (1) – (5) capture a set of standard assumptions about speaker/hearer modalities and common ground. In (3), we assume the presence of a negative introspection axiom, however, we do not require its full generality in practice.[3]

Axioms (6) – (10) specify a number of initial facts about speaker/hearer suppositions. In particular, (10) asserts a speaker supposition about common ground that illustrates the types of conclusions we typically require. These facts also include two K_v knowledge fluents, $K_v train$ and $K_v time$. As in PKS, these fluents act as placeholders for the values of known functions that can map to a wide range of possible values, but whose definite values may not be known at plan/reasoning time.

Rules (11) – (14) encode action precondition and effects axioms for two speech acts, *ask* and *tell*.

Using this axiomatization, we consider the task of constructing two dialog-based plans, as a problem of planning through proof.

4.4 Planning a Direct Speech Act

Goal: I need Bonnie to know which train I'll catch.

By speaker supposition, the hearer knows what time it is:

(15) $\Rightarrow [H]\, K_v time$ (*9b*); (*1*)

The speaker doesn't know what time it is:

(16) $\Rightarrow \neg [S]\, K_v time$ (*8b*); (*2*); (*7b*)

By speaker supposition, Bonnie doesn't know what train the speaker will catch:

(17) $\Rightarrow \neg [B]\, K_v train$ (*6b*); (*1*)

The speaker supposes it's not common ground with Bonnie as to what train the speaker will catch:

(18) $\Rightarrow [S] \neg [C_{SB}]\, K_v train$ (*8b*); (*2*); (*5*); (*3*); (*4*)

The situation affords $\mathit{ask}(S, H, K_v time)$:

(19) $\Rightarrow \mathit{affords}(\mathit{ask}(S, H, K_v time))$ (*16*); (*9b*); (*13b*)

After applying $\mathit{ask}(S, H, K_v time)$:

(20) $\Rightarrow [C_{SH}] \neg [S]\, K_v time$ (*19*); (*14b*)

The situation now affords $\mathit{tell}(H, S, K_v time)$:

(21) $\Rightarrow \mathit{affords}(\mathit{tell}(H, S, K_v time))$
(15); (20); (4); (5); (11*b*)

After applying $\mathit{tell}(H, S, K_v time)$:

(22) $\Rightarrow [S]\, K_v time$ (*21*); (*16*); (*12b*)

—which means I know what train I will catch:

(23) $\Rightarrow [S]\, K_v train$ (*22*); (*7b*)

The situation now affords $\mathit{tell}(S, B, K_v train)$

(24) $\Rightarrow \mathit{affords}(\mathit{tell}(S, B, K_v train))$ (*23*); (*18*); (*11b*)

After applying $\mathit{tell}(S, B, K_v train)$:

[3] The weaker property $[X] \neg p \Rightarrow [X] \neg [C_{XY}]\, p$ (which also follows from negative introspection) will typically suffice.

(25) $\Rightarrow$ [B] K_v*train* (*24*); (*17*); (*12b*)

4.5 Planning an Indirect Speech Act

The original situation also affords telling the hearer that I don't know the time:

(26) $\Rightarrow$ [S] $\neg$ [S] K_v*time* (*8b*); (*2*); (*7*); (*3*)

(27) $\Rightarrow$ [S] $\neg$ [C_{SH}] $\neg$ [S] K_v*time* (*10*)

(28) $\Rightarrow$ *affords*(*tell*(S, H, $\neg$ [S] K_v*time*)) (26); (27); (11*b*)

After saying "I don't know what time it is"—that is, applying the action *tell*(S, H, $\neg$ [S] K_v*time*),

(29) $\Rightarrow$ [C_{SH}] $\neg$ [S] K_v*time* (*14b*)

Since (29) is identical to (20), the situation again affords *tell*(H, S, K_v*time*), and the rest of the plan can continue as before.

Asking the time by saying "I don't know what time it is" would usually be regarded as an indirect speech act. Under the present account, both "direct" and "indirect" speech acts have effects that change the same set of facts about the knowledge states of the participants. Both involve inference. In some sense, there is no such thing as a "direct" speech act. In that sense, it is not surprising that indirect speech acts are so widespread: *all* speech acts are indirect in the sense of involving inference. Crucially, the plan does not depend upon the hearer identifying the fact that the speaker's utterance "I don't know what time it is" had the illocutionary force of a request or question such as "What time is it?".

From an axiomatic point of view, the above examples illustrate that the reasoning required to achieve the desired conclusions is straightforward—in most cases only direct applications of the domain axioms are used. Most importantly, we do not need to resolve knowledge-level conclusions like K_v*train* at this level of reasoning and, thus, do not require standard axioms of knowledge to reason about the formulae *within* the scope of $K/K_v/K_w$.

Direct manipulation of fluents like K_v*train* means that we can manage knowledge and sensing actions in a PKS-style manner in our account. For instance, the above plans result in the conclusion [S] K_v*time* as a consequence of the *ask* and *tell* actions. The particular effect of "coming to know the value" of *time* means that we should treat these actions as sensing actions. At the knowledge-level of abstraction, the effects of *ask* and *tell* are no different than the effect produced by reading a piece of paper to come to know a telephone number in our earlier example. This PKS-style use of knowledge fluents also opens up the possibility of constructing conditional plans and, ultimately, planning with PKS itself.

4.6 On So-called Conversational Implicature

The fact that we distinguish speaker suppositions about common ground from the hearer suppositions themselves means that we can include the following rules parallel to (11) and (12) without inconsistency:

(30) a. "X can always say p to Y"
b. $\Rightarrow$ *affords*(*say*(X, Y, p))

(31) a. "If X says p to Y, and Y supposes $\neg p$, then Y continues to suppose $\neg p$, and supposes that $\neg p$ is not common ground."
b. {*affords*(*say*(X, Y, p))} $\wedge$ [Y] $\neg p$
$\multimap$ [*say*(X, Y, p)][Y]$\neg p \wedge$ [Y] $\neg$ [C] $\neg p$

Speakers' calculations about what will follow from making claims about hearers' knowledge states extend to what will follow from making *false* utterances. To take a famous example from Grice, suppose that we both know that you have done me an unfriendly turn:

(32) a. "I know that you are not a good friend"
b. [S] $\neg$*friendship*(h) = *good*

(33) a. "You know that you are not a good friend"
b. [H] $\neg$*friendship*(h) = *good*

After applying *say*(S, H, *friendship*(h) = *good*), say by uttering the following:

(34) You're a fine friend!

the following holds:

(35) $\Rightarrow$ [H] $\neg$*friendship*(h) = *good*
$\wedge$ [H] $\neg$ [C] $\neg$*friendship*(h) = *good* (*32*); (*33*); (*31b*)

One might not think that getting the hearer to infer something they already know is very useful. However, if we assume a mechanism of attention, whereby things that are inferred become salient, then we have drawn their attention to their trespass. Moreover, the information state that we have brought them to is one that would normally suggest,

via rules like (11) and (12), that the hearer should tell the original speaker that they are not a fine friend. Of course, further reflection (via similar rules we pass over here) is likely to make the hearer unwilling to do so, leaving them few conversational gambits other than to slink silently and guiltily away. This of course is what the original speaker really intended.

4.7 A Prediction of the Theory

This theory explains, as Grice did not, why this trope is asymmetrical: the following is predicted to be an ineffectual way to make a hearer pleasantly aware that they have acted as a good friend:

(36) #You're a lousy friend!

It is counterproductive to make the hearer think of the key fact for themselves. Moreover, there is no reason for them not to respond to the contradiction. Unlike (34), this utterance is likely to evoke a vociferous correction to the common ground, rather than smug acquiescence to the contrary, parallel to the sheepish response evoked by (34).

5 Discussion

We have presented a number of toy examples in this paper for purposes of exposition: scaling to realistic domains will raise all the usual problems of knowledge representation that AI is heir to. However, the update effects (and side-effects) of discourse planning that we describe are general-purpose. They are entirely driven by the knowledge state, without recourse to specifically conversational rules, other than some very general rules of consistency maintenance in common ground. There is therefore some hope that conversational planning itself is of low complexity, and that any domain we can actually plan in, we can also plan conversations about.

According to this theory, illocutionary acts such as questioning and requesting are discourse subplans that are emergent from the general rules for maintaining consistency in the common ground and for manipulating knowledge-level information, such as the K_v formulae in our examples. Of course, for practical applications that require efficient execution, we can always memoize the proofs of such frequently-used sub-plans in the way that is standard in Explanation-Based Learning (EBL). For instance, by treating action sequences as "compound" actions in the planning process, we would be in effect compiling them into a model of dialog state-change of the kind that is common in practical dialog management. More importantly, the present work offers a way to derive such models automatically from first principles, rather than laboriously constructing them by hand.

In contrast to approaches that reject the planning model on complexity grounds, e.g., (Beun, 2001), our choice of a planner with limited reasoning capabilities and knowledge resources—conditions often cited as underlying human planning and dialog—aims to address such concerns directly. Furthermore, the specialized rules governing speech act selection in alternate approaches can always be adopted as planning heuristics guiding action choice, if existing planning algorithms fail to produce sufficient plans.

We have also argued that LDEC, extended with PKS-style knowledge primitives, is sufficient for planning dialog actions. Although we have motivated a correspondence between LDEC and PKS, we have not described how PKS planning domains can be formed from LDEC axioms. While some of the mechanisms needed to support a translation already exist—the compilation of LDEC rules into PKS queries and database updates is straightforward and syntactic—we have yet to extend PKS's inference rules to encompass speaker/hearer modalities, and formally prove the soundness of our translation. We are also exploring the use of PKS's *LCW* database to manage common ground as a form of closed world information. (For example, if a participant X cannot establish p as common ground then X should assume p is not common ground.) Finally, we require a comprehensive evaluation of our approach to assess its feasibility and scalability to more complex dialog scenarios. Overall, we are optimistic about our prospects for adapting PKS to the problem of planning dialog acts.

Acknowledgements

The work reported in this paper was partially funded by the European Commission as part of the PACO-PLUS project (FP6-2004-IST-4-27657), and by the NSF under grant number NSF-IIS-0416128.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press, Cambridge.

Robbert-Jan Beun. 2001. On the generation of coherent dialogue. *Pragmatics and Cognition*, 9:37–68.

Wolfgang Bibel, Luis Farinas del Cerro, B. Fronhfer, and A. Herzig. 1989. Plan generation by linear proofs: on semantics. In *German Workshop on Artificial Intelligence - GWAI'89*, volume 216 of *Informatik-Fachberichte*, Berlin. Springer Verlag.

Michael Bratman, David Israel, and Martha Pollack. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355.

Jennifer Chu-Carroll and Sandy Carberry. 1995. Response generation in collaborative negotiation. In *Proceedings of ACL-95*, pages 136–143. ACL.

Philip Cohen and Hector Levesque. 1990. Rational interaction as the basis for communication. In Philip Cohen, Jerry Morgan, and Martha Pollack, editors, *Intentions in Communication*, pages 221–255. MIT Press, Cambridge, MA.

Robert Demolombe and Maria del Pilar Pozos Parra. 2000. A simple and tractable extension of situation calculus to epistemic logic. In *Proceedings of ISMIS-2000*, pages 515–524.

Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. 1992. An approach to planning with incomplete information. In *Proceedings of KR-92*, pages 115–125.

Oren Etzioni, Keith Golden, and Daniel Weld. 1994. Tractable closed world reasoning with updates. In *Proceedings of KR-94*, pages 178–189. Morgan Kaufmann Publishers.

Richard Fikes and Nils Nilsson. 1971. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Jean-Yves Girard. 1987. Linear logic. *Theoretical Computer Science*, 50:1–102.

Nancy Green and Sandra Carberry. 1994. A hybrid reasoning model for indirect answers. In *Proceedings of ACL-94*, pages 58–65. ACL.

Barbara Grosz and Candace Sidner. 1990. Plans for discourse. In Philip Cohen, Jerry Morgan, and Martha Pollack, editors, *Intentions in Communication*, pages 417–444. MIT Press, Cambridge, MA.

David Harel. 1984. Dynamic logic. In Dov Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. Reidel, Dordrecht.

Robert Kowalski and Maurice Sergot. 1986. A logic-based calculus of events. *New Generation Computing*, 4:67–95.

Lynn Lambert and Sandra Carberry. 1991. A tripartite plan-based model of dialogue. In *Proceedings of ACL-91*, pages 47–54. ACL.

Diane Litman and James Allen. 1987. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200.

Colin Matheson, Massimo Poesio, and David Traum. 2000. Modeling grounding and discourse obligations using update rules. In *Proceedings of NAACL 2000, Seattle*.

Nicolas Maudet. 2004. Negotiating language games. *Autonomous Agents and Multi-Agent Systems*, 7:229–233.

John McCarthy and Patrick Hayes. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence*, volume 4, pages 473–502. Edinburgh University Press, Edinburgh.

Robert Moore. 1985. A formal theory of knowledge and action. In Jerry Hobbs and Robert Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, Norwood, NJ. Reprinted as Ch. 3 of (Moore, 1995).

Robert Moore. 1995. *Logic and Representation*, volume 39 of *CSLI Lecture Notes*. CSLI/Cambridge University Press, Stanford CA.

Leora Morgenstern. 1988. *Foundations of a Logic of Knowledge, Action, and Communication*. Ph.D. thesis, NYU, Courant Institute of Mathematical Sciences.

Ronald P. A. Petrick and Fahiem Bacchus. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of AIPS-02*, pages 212–221.

Ronald P. A. Petrick and Fahiem Bacchus. 2004. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proc. of ICAPS-04*, pages 2–11.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artifial Intelligence Research*, 16:105–133.

Mark Steedman. 1997. Temporality. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 895–938. North Holland/Elsevier, Amsterdam.

Mark Steedman. 2002. Plans, affordances, and combinatory grammar. *Linguistics and Philosophy*, 25:723–753.

Mark Steedman. 2006. Surface compositional semantics of intonation. *In submission*.

Matthew Stone. 1998. Abductive planning with sensing. In *Proceedings of AAAI-98*, pages 631–636, Menlo Park CA. AAAI.

Matthew Stone. 2000. Towards a computational account of knowledge, action and inference in instructions. *Journal of Language and Computation*, 1:231–246.

David Traum and James Allen. 1992. A speech acts approach to grounding in conversation. In *Proceedings of ICSLP-92*, pages 137–140.

R. Michael Young and Johanna D. Moore. 1994. DPOCL: a principled approach to discourse planning. In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 13–20.