Toward Domain-Independent Dialogue Planning

Tânia Marques and Michael Rovatsos

University of Edinburgh
Edinburgh EH8 9AB, United Kingdom
{tmarques,mrovatso}@inf.ed.ac.uk

Abstract. While the development of techniques that allow artificial agents to engage in dialogue with humans has received a lot of interest in the multiagent systems and natural language processing literature, most of the systems created to date have focused on specific domains and types of dialogue. This has led to agent designs that are useful for specific dialogue situations, but hard to adapt to different settings. The creation of more flexible agents that can deal with a broad range of communicative scenarios would greatly improve the interaction between agents and humans, and would eliminate the need to manually adapt the design of a conversational agent when dealing with a new task domain. In this paper, we present initial work toward creating agents that are able to generate task-oriented dialogues based on a description of a previously unknown domain. Our method is based on utilising automated planning methods, which are suitable for processing specifications both of the communication language to be used and of the domain in hand. We provide preliminary experimental results which suggest that our method has the potential to provide the flexibility required to produce a broad range of communication behaviours in different settings.

Keywords: Dialogue Planning, Conversational Agents, Human-Agent Dialogue

1 Introduction

Whenever agents situated in the same environment need to coordinate their activities, explicit communication is a useful for anticipating and influencing others' behaviours. Such communication is often organised using dialogue as a means to structure complex exchanges of messages, e.g. to implement procedures for information seeking, negotiation, the monitoring of joint activity, etc.

While multiagent systems research has produced a rich body of work in terms of designing expressive and versatile agent communication languages [1] and interaction protocols [2], there have been very few attempts to design agents that can rationally plan their communication behaviour when the communication language, interaction protocol, and task domain specification are only provided at *runtime*. At the same time, this kind of flexibility is particularly important when artificial agents interact with human users, as it is difficult to anticipate the communicative actions of humans on the one hand, and tedious to re-implement dialogue agents for every different problem domain on the other.

In the area of human-agent dialogue, sets of domain-specific communicative actions have to be specified manually by the designer, or learnt from data [3, 4]. This is a useful approach in the sense that it allows the creation of versatile dialogues for that specific situation. Nevertheless, it means that the agents have to be re-programmed or re-taught to be able to communicate in another domain, even when dealing with very similar situations. Dropping this assumption would enable us to develop more flexible agents that can be easily adapted to different task domains.

Automated dialogue generation from a task domain description is a challenging task, as it is hard to determine appropriate dialogue actions when the domain is not specified *a priori*. More specifically, this is due to the fact that, even though specifications of communicative acts define the syntactic structure of utterances and the semantic constraints that govern their use, this still normally allows for a huge set of possible concrete messages in any given situation. For example, a simple information exchange consisting of a question *"Could you help me perform action X?"* and a subsequent response *"I can help you by doing Y if you do Z for me in return"* allows for a very large number of choices of instance values for X, Y, and Z in any realistic domain.

To address this problem, we use symbolic planning techniques [5], as they afford us with domain-independent methods of determining action sequences in a goal-oriented way, so that the agent's objectives drive the process of communication, and narrow down communicative choices to those that are expected to further the agent's current goals. More specifically, we use a two-layered planning approach, where plan-based representations and inference are used both for determining the right course of action regarding the achievement of task objectives *and* for making communicative decisions. Our dialogue planning agents employ planning at two levels:

– **Domain Planning:** The agent comes up with a plan to solve its task based on the domain description provided. This domain plan is subsequently used to create communicative intentions and desires, whenever there is a goal that cannot be achieved without actions being performed by another agent. These actions may be actual actions of the domain, or may indicate the need to ask for further information due to the existence of unknown facts.
– **Communicative Planning:** The agent uses communicative intentions, desires, and a set of given communicative act specifications to come up with a plan that describes what to say to the other agent in order to achieve its task. Communicative plans involve expected responses from interlocutors, and will obviously have to be revised during execution, depending on the responses actually received.

In terms of uncertainty, symbolic planning obviously implies a more coarse-grained approach than probabilistic approaches (the agent can only distinguish between known and unknown facts, rather than quantify different degrees of belief). Also, our approach currently assumes task-oriented dialogues that are driven by a finite set of all-or-nothing goals. In return for these limitations, symbolic planning provides us with representations that can be easily combined with the semantics of speech acts commonly used in agent communication languages [6], and makes it easier to bootstrap decision-making algorithms without domain-specific quantitative expectations about the likely behaviours of other agents.

Apart from describing the design of our planning-based dialogue agent, this paper presents preliminary results obtained with an implementation of our method in two different *domains* not known to the agent *a priori*, illustrating its adequacy for cross-domain use. In the longer term, we envision that similar agent algorithms could also process different *language specifications*, so that they could also automatically switch between completely different communication contexts, and the work presented in this paper is an initial step toward this long-term goal.

The remainder of the paper is structured as follows: We start by discussing related work in Section 2. Our two-layer dialogue planning architecture is described in Section 3. Section 4 presents initial results obtained with an implementation of our dialogue agents in two different task domains. Section 5 concludes.

2 Related Work

The development of methods to structure and synthesise dialogue that involves artificial agents has received much attention across various communities over the past thirty years.

In the agents community, much work has gone into defining agent communication standards, e.g. through the FIPA initiative [7], which proposed standards for speech-act based communication languages that specify a number of primitives an agent may use (inform, request, etc). While research produced in this area [1] involves developing appropriate formal models for the syntax and semantics of the languages and protocols involved, these are defined with interoperability in mind. This means that the focus is on defining *minimal* models of communication semantics, rather than describing how their specifications could actually be used for making communication decisions. Our work borrows much from this work, but extends the definitions of speech acts with additional information that makes them appropriate for performing concrete dialogue planning activities. This is very much in the spirit of early work on planning with speech acts [6], although we take a much more practical approach with a focus on implementation of actual dialogue generation architectures and algorithms.

This, of course, is a direction that has already been explored in previous work. In the COLLAGEN system [8], each agent creates a prioritized agenda of the possible communicative actions that might contribute to the current dialogue. Then, the system generates the sentences that correspond to the action that has the highest priority using a set of templates. Moore and Paris [9] propose a descriptive theory to plan text generation based on relations among parts of the text (e.g. persuasion, motivation, etc). Both these works, however, focus on communication planning while disregarding the connection between the domain task and the specification of the communication language. In other words, the process of constructing dialogue actions to achieve some goal from an individual agent's perspective is not accounted for in these models. Considering this problem, Lambert and Carberry [10] propose a tripartite model of dialogue that distinguishes between domain, problem solving and discourse or communicative actions. They, too, assume that there is shared knowledge about the discourse and problem-solving plans among the agents, however. Sklar et al. [11] propose a model of human-robot interaction, similar to ours, where the robot is equipped with a set of beliefs in a logical representation that allows it to infer what to do. When the robot is faced with an action that requires human assistance, it initiates a dialogue that follows one of their predefined protocols. This differs from our approach, because their argumentation steps are based on pre-defined protocols rather than being planned as a function of the domain plan. Differently from all the authors above, who assume that the agents communicate to construct a domain plan, we are interested in agents that are trying to perform their own task and discover autonomously when to communicate or if they need to communicate at all. This means that the existence of a shared domain plan is not assumed. Instead, we explore how agents can collaborate over and exchange information about parts of their individual plans that may overall, or where there is flexibility regarding the choice of plan that can achieve an individual agent's goals.

Further existing work focuses mainly on making optimal communication decisions in a very specific task domain. For example, Negochat [3] is a negotiation agent that communicates very effectively using a natural language chat interface in scenarios where there is bilateral negotiation and two agents aim to achieve agreement on conflicting issues. This differs from our approach in that our focus is on agents being able to perform several different communicative acts such as negotiating, asking for information, and so on, while covering as many different domains as possible. Since we are only interested in the overall achievement of goals, we do not focus on reward functions and optimal communication behaviour in a decision- or game-theoretic sense.

Within the field of conversational service robots, robotic agents have to plan their dialogue to interact with human users. For instance, Giuliani et al. [12] presented a bartender robot that used symbolic planning to communicate appropriately with a human user based on its overall task and on social norms. Despite the use of symbolic planning methods, their approach was specifically tailored to the domain of robot bartending. A more generic robot design was proposed by Nakano et al. [13], which can switch the domain it is talking about when a human wants to perform a different task. For this, they use a two-layer model: A global task planner that uses hierarchical planning methods, and a lower layer that involves a local planning base with expert modules for specific physical actions and for engaging in dialogue. Their approach is similar to ours, even though the goal of the robot is, in their case, to converse with human users in specific known domains, while we want to create domain-independent agents that focus on their own task and communicate only when they consider it necessary for them to achieve their own goals.

3 Two-Layer Dialogue Planning

In order to be capable of dealing with different task domains, a dialogue agent needs to be able to generate utterances by processing specifications of a possible communicative action schemata (and constraints on their usage in certain dialogue protocols or conversation models) and of the task domain *at runtime*. This processing needs to be organised in such a way that it allows the agent to decide what the most suitable next utterance might be in terms of advancing the achievement status of its local goals. We propose a two-level planning approach to tackle this problem. At the lower level, our agent looks at the domain and task description, and creates a domain plan for solving the task based on its (usually limited) knowledge of the world. This plan may contain actions that the agent cannot perform on its own, including "joint" actions (e.g. "moving a table together"), actions only the other agent is capable of performing, and actions that require exchange of resources between agents. Below, we refer to this type of actions as *collaborative actions*. The domain plan may also contain artificially generated actions that convert unknown values to concrete ones, and which can only be executed if the real value of the unknown fact has been confirmed. At the higher level, the agent looks at its beliefs of the domain and its communicative goals, obtained from the domain plan, to generate an adequate utterance that will allow it to reach its communicative goal (and thus, if the utterance has the desired effect, its domain goal). If the utterance leads to unexpected or undesirable effects, the agent will attempt to replan in order to determine appropriate subsequent actions based on its revised beliefs about the situation.

In what follows, we will assume that agents are self-interested though willing to collaborate if the collaborative acts do not jeopardize their own goals. We also assume that the agents are honest, i.e. they never provide false information. Finally, we assume that an utterance changes the internal state of an interlocutor in exactly the same way as specified by the communicative act definition, which is assumed to be common knowledge among interlocutors. While these are strong assumptions, they could be relaxed by creating a more complex model of the other agent that is updated according to what it utters and the implications of such utterances. This is a direction we intend to pursue in the future.

3.1 Planning Algorithms and Representations

In the context of classical STRIPS-style single-agent planning [5], a planning problem $\Pi = \langle F, A, I, G \rangle$ consists of a finite set of *fluents* F, i.e. logical propositions used to describe the current state of the world and *action schemata* $a = \langle pre, eff \rangle \in A$ with *preconditions* pre (which have to be satisfied for a to be feasible) and *effects* eff which specify which fluents become true/false after a is performed. The action $move(agent_A, position_A, position_B)$, for example, moves $agent_A$ from $position_A$ to $position_B$, and has the fact that $agent_A$ is in $position_A$ as its precondition, and that $agent_A$ is no longer in $position_A$ but has moved to $position_B$ as its effects. The *initial state* I is specified by a set of fluent propositions currently true, and the *goal specification* G is a set of propositions that have to be achieved by a *plan* $\pi = \langle a_1, \ldots a_n \rangle$ from I such that the resulting state satisfies all propositions in G. Action schemata may contain variables, but these are normally assumed to range over finite domains, so that every "first-order" schema can, in principle, be replaced by a finite number of ground actions.

Planning can be conceptually viewed as a search problem in a state space where the connections between states are established according to the preconditions and effects of the actions that are applicable in that state. In this conceptualization, a plan is a path that leads from the initial state to one of the goal states. Optimal planning attempts to find a cost-minimal plan that solves the problem, while satisficing planning may return any solution.

Whenever we refer to planning processes below, this will imply the use of a satisficing classical planner (Metric-FF [15], in our implementation) from an individual agent's point of view, given a planning problem that may also involve other agents' actions. Note that while the agent may be able to plan other agents' actions hypothetically, she is not able to cause them to happen.

To represent the planning domain, we use the Planning Domain Definition Language (PDDL) [19] as a standard used by most common planning algorithm implementations. PDDL specifications include a domain description file (which specifies the action schemata for a domain) and a problem file (which defines a specific instance of a planning problem). However, our domains require certain information that is usually not represented in PDDL, such as the definition of which actions are collaborative, a definition of the other agent's goals, and a facility to allow some values of the domain to be unknown. To avoid loss of compatibility with off-the-shelf planners, we specify this information in a format that complies with the standard PDDL syntax and semantics. Collaborative actions and goals are designated by special predicates

```
(COLLABORATIVE action_name)
```

and

```
(GOAL agent_id goal_predicate)
```

which assumes that all actions, agent names, and goal predicates are "reified" as objects in the domain. How these meta-predicates are used will be explained below, when we describe the details of how communication planning takes collaborative actions and assumptions about other agents' goals into account.

Expressing that certain values are unknown requires using a special `unknown` parameter inside domain predicates:

```
(PREDICATE_NAME parameter_1 ... unknown ... parameter_n)
```

The reason for allowing the agent to deal with "known unknowns", i.e. aspects of the environment which it has no information about (while being aware of this uncertainty), is that it enables us to deal with domains in which some information may not be available to one agent, but might be obtained from the other agent.

This could be dealt with by simply generating questions asking other agents for their actual values prior to trying to synthesise an appropriate domain plan. This is, however, computationally wasteful as it may be completely unnecessary in many situations. Instead, we follow a strategy that identifies which unknown values may be useful for solving the task in hand. More specifically, every time the system identifies a predicate instance with unknown values, it generates domain actions that convert the unknown value to all its possible values. Then, during domain planning, the planner may use some of those actions if needed. If the plan contains such an action, the agent knows that it must ask somebody else about the value of this "unknown". This approach restricts exploration of unknowns to those values that might be of interest.

Note that this strategy benefits from the "greedy" nature of search-based planning algorithms. Also, we should remark that for state-of-the-art planners, the seemingly wasteful generation of all possible instance values for unknown parameters does not pose serious increases in planning times in all but the most complex domains (these algorithms actually regularly instantiate any parameterised action schemata as part of their pre-processing as a matter of course).

3.2 Communicative Action Schemata

Essential to our approach is coming up with a method of specifying possible utterances in such a way that an agent can reason about them in terms of preconditions and postconditions, so that standard planning methods can be applied to decide what to say next.

This means that a message is equivalent to a belief transfer induced by the communicative acts, and the content of the message consists of filling the variables in the specification of a communicative action schema with actions and beliefs relevant in the domain.

In our current implementation, the communicative world involves four types of entities: agents *Ag*, beliefs *Bels*, actions *Acts*, questions *Ques* and answers *Ans*. All actions are represented as actions in the planning domain. Questions *Ques* correspond to facts unknown to the agent, and which it could ask its interlocutor about. When an agent receives a question, the known facts that could be returned are the elements of *Ans*, i.e. those found in its own knowledge base that match the question received.

In addition to beliefs about domain facts, which are represented as planning propositions, dialogue agents hold several types of additional beliefs regarding mental states, capabilities, and dependencies between questions and answers:

- $des(a, act)$ – agent $a \in Ag$ desires action $act \in Acts$;
- $cap(a, act)$ – agent a is capable of performing action act;
- $int(a, act)$ – agent a intends to perform action act;
- $cond\_int(a, act_1, act_2)$ – agent a intends to perform action act_1 if action act_2 is performed;
- $no\_int(a, act)$ – agent a does not intend to perform action act (required to avoid using negation; STRIPS-style planning assumes "negation as failure", so a negated fact can never be a goal);
- $unknown(a, que)$ – the answer to question que is unknown to agent a;
- $bel\_ans\_que(a, ans, que)$ – agent a's answer ans answers question que; and
- $unwilling\_perform(a, act)$ – agent a is not willing to perform action act.

Furthermore, a number of control predicates are used to track which actions have been performed before, so that using these in action preconditions and effects ensures constraints regarding admissible sequences of sentences are respected, and correspond to the rules normally specified in an interaction protocol:

- $proposal(a, act)$ – agent a has proposed action act;
- $proposed(a, b)$ – agent a has made a proposal to agent b (ensures that no new proposals are made until a previous proposal has been responded to);
- $question(a, que)$ – agent a has asked question que;
- $asked(a, b)$ – agent a has asked a question to agent b (ensures a question is answered before another one is asked);
- $answered(a, b, que)$ – agent a has answered agent b's question que.

With these provisions, we are now able to provide a first simple set of communicative acts or a communicative action schemata which is based on speech acts: *propose, proposeIf, accept, reject, ask* and *reply.* These communicative acts are defined as ordinary planning actions.

- $\mathbf{propose(a, b, act)}$
  [agent a proposes that agent b perform action act]
  ::preconditions $a = me \wedge des(a, act) \wedge cap(b, act) \wedge \neg cap(a, act)$
  $\wedge \neg proposed(a, b)$
  ::effects $proposed(a, b) \wedge proposal(a, act)$
- $\mathbf{proposeIf(a, b, act1, act2)}$
  [a proposes to b that she will perform $act2$ if b performs $act1$]
  ::preconditions $a \neq b \wedge des(a, act1) \wedge \neg cap(a, act1) \wedge ((des(b, act2)$
  $\wedge \neg cap(b, act2)) \vee (proposed(b, a) \wedge proposal(a, act2)))$
  $\wedge \neg unwilling\_perform(a, act2))$
  ::effects $proposed(a, b) \wedge proposal(a, act1) \wedge cond\_int(a, act2, act1)$
  $\wedge \neg proposed(b, a) \wedge \neg proposal(b, act2)$

- $\mathbf{accept-proposal(a, b, act)}$
  [a accepts b's proposal that a should perform act]
  ::preconditions $a \neq b \wedge \mathit{proposed}(b, a) \wedge \mathit{proposal}(b, act)$
  $\wedge \neg \mathit{unwilling\_perform}(a, act)$
  ::effects $\mathit{int}(a, act) \wedge \neg \mathit{proposed}(b, a) \wedge \neg \mathit{proposal}(b, act)$
- $\mathbf{reject-proposal(a, b, act)}$
  [a rejects b's proposal that a should perform act]
  ::preconditions $a \neq b \wedge \mathit{proposed}(b, a) \wedge \mathit{proposal}(b, act)$
  $\wedge \mathit{unwilling\_perform}(a, act)$
  ::effects $\mathit{no\_int}(a, act) \wedge \neg \mathit{proposed}(b, a) \wedge \neg \mathit{proposal}(b, act)$
- $\mathbf{ask(a, b, que)}$
  [a asks b question que]
  ::preconditions $a = me \wedge \mathit{unknown}(a, que) \wedge \neg \mathit{asked}(a, b)$
  $\wedge \neg \mathit{answered}(b, a, que)$
  ::effects $\mathit{asked}(a, b) \wedge \mathit{question}(a, que)$
- $\mathbf{reply(a, b, que, ans)}$
  [a replies to b's question que with answer ans]
  ::preconditions $a \neq b \wedge \mathit{asked}(b, a) \wedge \mathit{question}(b, que)$
  $\wedge \mathit{bel\_ans\_que}(a, ans, que) \wedge \neg \mathit{answered}(a, b, que)$
  ::effects $\mathit{answered}(a, b, que) \wedge \mathit{bel\_ans\_que}(b, ans, que) \wedge \neg \mathit{unknown}(b, que)$

These communicative acts are defined for any $a, b \in Ag$, $act, act1, act2 \in Acts$, $ans \in Ans$, $que \in Que$, and a special constant me is used to refer to the identifier of the planning agent herself. With the interpretation of *not* ($\neg$) as "false if predicate not present in state specification" (negation as failure), and disjunction in preconditions "compiled away" in the pre-processing stage of any standard classical planner, these definitions remain within the expressiveness of PDDL. It is important to emphasise, once more, that these speech acts only capture scenarios where the dialogue is task-oriented. They would not be sufficient for dealing with other scenarios such as enquiry, argumentation, or advice-giving.

3.3 Dialogue architecture

The dialogue architecture, as shown schematically in figure 1, describes the overall control flow for the dialogue agent, both for the *speaker* and the *hearer* role. In the speaker role, the agent first creates a domain plan, plans any relevant communication actions, and communicates them to the hearer. Domain plan creation is triggered by determining whether a plan that achieves the agent's goal can be performed by the agent herself. If there are collaborative actions in the plan, the agent generates a potential joint plan where both agents achieve their goal, which requires that the speaker makes some assumption about the hearer's possible goals. The agent translates the domain plan into beliefs $bels \subseteq Bels$ about the mental states, capabilities and desires of both the speaker and the hearer as defined in section 3.2. The communicative goals G_{comm} are a subset of those beliefs that indicate what this agent wants the other agent to come to believe. These may involve action intentions ($\mathit{int}(a, act)$), negative action intentions ($\mathit{no\_int}(a, act)$), or intentions to answer a question ($\mathit{answered}(a, b, que)$) (algorithm 1 explains how these are generated). For collaborative actions needed to achieve the domain goal, the agent generates an intention. To deal with unknown values that could be required to perform a plan, the agent generates *a*

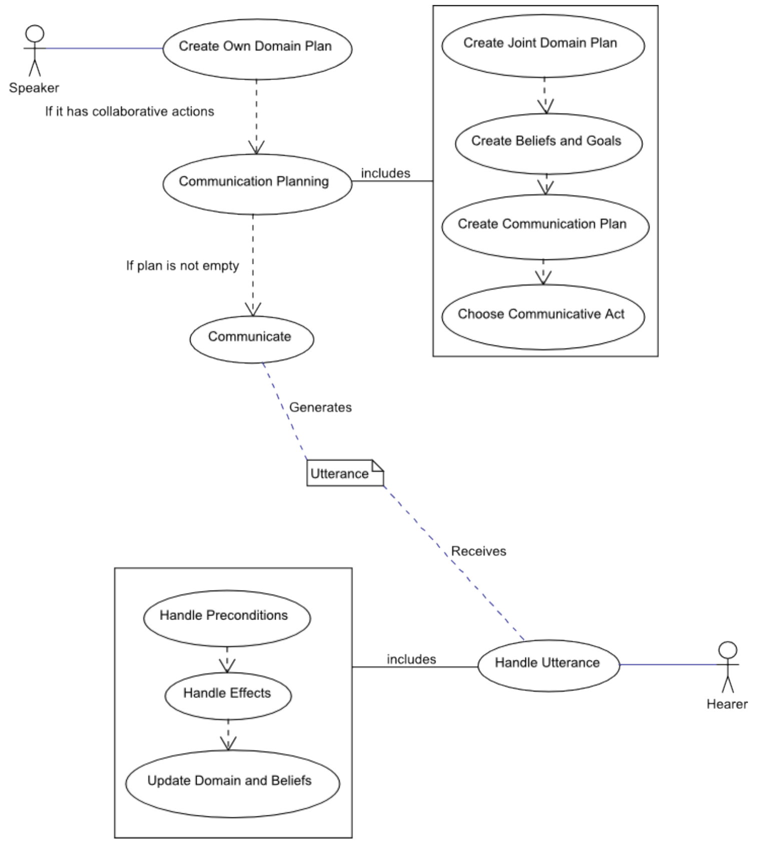


Fig. 1. Overview of dialogue architecture emphasizing the two roles that the agents may assume: speaker and hearer. Any agent may initiate the interaction, and then it proceeds in a strictly turn-taking fashion.

priori a number of artificially constructed actions to convert the unknown values into each possible combination. Each of these artificially added actions is converted to a question, with the intention being adopted that it be answered. Furthermore, the system also creates beliefs regarding the action in terms of desirability and capability, which depends on whether the action is collaborative or not, and who can perform it.

Based on these beliefs, the planner is invoked to obtain a communication plan consisting of instances of the communicative action schemata, from which the first action will be performed, and sent to the hearer as a message. After communicating or receiving an utterance, the communicative act contained in it triggers the respective effects both in the communicative and domain description, which leads to an update of beliefs at both levels. This may correspond, depending on the communicative act, to an action being performed by one of the agents in the domain. This process is shown in algorithm 2.

Algorithm 1 CreateBeliefsAndGoals function

```
Input: $plan$ - domain plan that fulfils both agents' goals; $bels$ - this agent's beliefs; $G_{comm}$ - own
  communicative goals; $Col_{act}$ - collaborative actions;
Output: $bels$ - updated beliefs; $G_{comm}$ - updated communicative goals
  for $act \in plan$ do
    {In the case of artificially constructed actions with "unknown" preconditions, generate a goal
    to have a question about them answered by the other agent.}
    if $unknown \in getPreconditions(act)$ then
      $que$ = predicate with $unknown$ value
      add $answered(other\_agent, me, que)$ to $G_{comm}$
    end if
    if $act \in Col_{act}$ then
      {$agent_i$ is the agent who performs $act$ and $agent_j$ the agent who benefits from it}
      add $des(agent_j, act)$ to $bels$
      add $capable(agent_i, act)$ to $bels$
      add $\neg capable(agent_j, act)$ to $bels$
      if $act$ in contributes to own goal, and not to the other's goal then
        add $int(other\_agent, act)$ to $G_{comm}$
      end if
    else
      add $des(agent_i, act)$ to $bels$
      add $capable(agent_i, act)$ to $bels$
    end if
  end for
```

In the hearer role, the agent's behaviour is controlled by the function shown in algorithm 3. First, the agent looks at the preconditions of the received utterance, which should be true if it was used correctly by the other agent. If these preconditions go against the hearer's beliefs, then it will consider them false beliefs held by the speaker. Otherwise, the hearer will delete the contradictory beliefs. The agent also looks at the effects of the utterance, which should reflect the intended effect of the perceived message. These effects are added to the beliefs of the agent and, they also lead to a revision of its own beliefs and an update of the communicative goals: The agent either achieved her communicative goal, or it may have become unachievable.

If the effect involves a new intention or an answer from the other agent about some unknown value, the agent will update its knowledge of the domain. If the effects relate to a proposal or a question, then the agent will accept the proposal if it does not jeopardize its goals, and answer the question if it is able to do so. The update of communicative goals and the revision of the cognitive state model of the other agent will be used as input to re-planning processes at the domain and communicative level. The hearer will re-plan right away, while the speaker will wait a certain time until generating a new sentence if no response is received, or become a hearer after receiving another utterance.

It is worth noting that the algorithm assumes speech act semantics are applied truthfully by both agents in our implementation of these algorithms. Different designs would be necessary it this assumption were relaxed, which would involve more complex forms of reasoning about the agent's own behaviour and that of her interlocutor.

4 Examples

In this section, we present examples from initial experiments obtained with our prototype to illustrate its capacity to generate dialogues across different domains.

Algorithm 2 Speaker algorithm

```
Input: $Acts$ - domain actions; $S_0$ - initial state; $S_{g0}$ - own goal; $S_{g1}$ - other agent's goal; $Col_{act}$ -
  collaborative actions
Output: $ut$ - utterance (instance of a communicative action schemata)
  $plan$ =createOwnDomainPlan($Acts$, $S_0$, $S_{g0}$)
  if $\exists act \in plan.(act \in Col_{act})$ then
    $joint_{p}lan$ = createJointDomainPlan($Acts$, $S_0$, $S_{g0}$, $Col_{act}$, $S_{g1}$)
    $bels, G_{comm}$ = createBeliefsAndGoals($joint_{p}lan$, $bels$, $G_{comm}$, $Col_{act}$)
    $plan_{comm}$ =createCommunicationPlan($Acts_{comm}$, $bels$)
    $ut$ = chooseFirstCommunicativeAct($plan_{comm}$)
    communicate ($ut$)
    updateDomainAndBeliefs($ut$, $G_{comm}$, $Acts_{comm}$, $bels$)
  end if
```

We chose two domains, the *Kit Delivery domain* [20], which does not normally involve communication, and the Colored Trails domain [21], which is commonly used for experiments that involve communication and planning. By introducing communication in the former scenario, we show that our method can be used to extend planning domains with communicative actions. By introducing partial observability in the second scenario, we demonstrate how our framework can easily accommodate situations that involve "known unknowns".

In the Kit Delivery domain, a fleet of robots must travel on a pre-defined path and gather a list of parts to assemble a kit with the constraints that each robot can take only two kit boxes at a time and they cannot overtake each other. In our example, we use two agents, two kits and seven locations, and we assume full observability. Additionally, we introduce the constraint that only one of the robots (say, agent 1) is able to deliver the kits at the target location. Thus, the other agent (agent 0) needs to communicate and ask the other agent to deliver the kit. To account for this, all we have to do is supply agent 0 with the information that the action involving kit delivery and return to the starting point (DELIVER_KIT_AND_RESTART) is a collaborative action. Agent 1 has delivering kit 2 as its goal, while agent 0 wants to deliver kit 1.

Using our architecture, agent 0 generates a plan that consists of picking kit 1 up at the starting point, taking it to the end point where agent 1 is located, handing over the kit, and agent 1 delivering it. Given that the final action is collaborative, the following dialogue ensues (we present utterances in human-readable format, with the actually generated dialogue move given below each utterance):

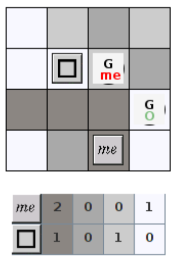
Agent 0: I propose that you deliver kit 1 for me.
(PROPOSE DELIVER_KIT_AND_RESTART AGENT1 KIT1 PLACEFORKIT2A)

Agent 1: I accept the proposal to deliver kit 1.
(ACCEPT-PROPOSAL DELIVER_KIT_AND_RESTART AGENT1 KIT1 PLACEFORKIT2A)

This is a very simple dialogue, but it shows that agent 0 recognizes it needs to communicate, and generates an appropriate proposal to the other agent to achieve the action it required. Apart from annotating a domain action as being collaborative, enabling such dialogue does not require any modifications to the original planning domain, and we were able to simply add the communicative action schemata and dialogue algorithm to the agent's design to produce a correctly executed dialogue.

Algorithm 3 Hearer Algorithm

```
Input: $Acts_{comm}$ - communicative action schemata; $ut$ - utterance; $bels$ - this agent's beliefs;
  $G_{comm}$ - this agent's communicative goals
  $preconditions$ = getPreconditions ($ut$, $Acts_{comm}$)
  for $pre \in preconditions$ do
    if $pre$ contradicts $bels$ then
      {The precondition contradicts the hearer's beliefs}
      if $pre$ is about itself then
        {The other agent is mistaken given the hearer's own beliefs about itself}
        add $pre$ to list of the other agent's false beliefs
      else
        {This agent is mistaken about the other agent}
        remove contradictory facts from $bels$
      end if
    else
      add $pre$ to $bels$
    end if
  end for
  $effects$ = getEffects($ut$, $Acts_{comm}$)
  for $eff \in effects$ do
    remove $eff$ from $G_{comm}$
    add $eff$ to $bels$
    {If there is a belief that contradicts this effect, then remove it from the agents' beliefs}
    if $eff$ contradicts $bels$ then
      remove contradictory facts from $bels$
    end if
    {If the effect refers to an intention of the other agent, update state}
    if $eff = int(other\_agent, act_x)$ then
      update state with effects of $act_x$
      {In case of a conditional intention of the other agent to perform $act_x$ in return for $act_y$,
      keep it in a temporary list of conditional intentions}
      if $eff = cond\_int(other\_agent, act_x, act_y)$ then
        add $cond\_int(other\_agent, act_x, act_y)$ to list of conditional intentions
      end if
      {If $act_x$ is in the list of conditional intentions, remove it and consider it fulfilled}
      if $act_x \in$ list of conditional intentions then
        remove $cond\_int(other\_agent, act_x, act_y)$ from the list of conditional intentions
        update state with effects of $act_y$
      end if
    end if
    {If the agent will not perform $act_x$, make it unusable for the current planning process}
    if $eff = no\_int(other\_agent, act_x)$ then
      block $act_x$ for future planning by adding a $false$ precondition to it
    end if
    {If the utterance implies that the other agent believes $ans_x$ is the answer for $que_x$, modify
    domain description accordingly}
    if $eff = bel\_ans\_que(this\_agent, ans_x, que_x)$ then
      replace $que_x$ by $ans_x$ in the domain description
    end if
    {If other agent proposed $act_x$ and hearer not able to perform this action, add a $no\_intention$
    to the communicative goals.}
    if $eff = proposal(other\_agent, act_x)$ then
      if cannot perform $act_x \in$ then
        add $no\_int(this\_agent, act_x)$ to $G_{comm}$
      end if
      {Add an intention of performing the action $act_x$ if the action is part of the joint plan, or if
      it is possible to still reach the goal after performing it.}
      if $act_x$ part of current plan or does not affect its performance then
        add $int(this\_agent, act_x)$ to $G_{comm}$
      else
        add $no\_int(this\_agent, act_x)$ to $G_{comm}$
      end if
    end if
    {If the effect corresponds to a question that was asked, then the agent searches for an answer
    to the question in its knowledge base and creates a communicative goal to answer the other
    agent.}
    if $eff = question(other\_agent, que_x)$ then
      return $ans_x$ that matches $que_x$ if contained in own knowledge base
      add $answered(this\_agent, other\_agent, que_x)$ to $G_{comm}$
    end if
  end for
```



me	2	0	0	1
◻	1	0	1	0

Fig. 2. The Colored Trails interface showing our example problem instance. The top part shows the playing board with locations of the reasoning agent ("me") and the opponent (denoted by a square), as well as goal locations labelled with "G". The bottom part shows the number of chips every agent currently has for each chip colour.

Note that in this particular example there exist more optimal plans than the one constructed by agent 0, e.g. a plan where agent 0 hands the kit to agent 1 at the starting point, since agent 1 is going to go there to pick up its own kit, and can carry two kits to the destination. Which of the possible plans will be suggested depends both on the planner used, and on the "mental theory" embedded in a concrete design of communicative acts (which involves making assumptions about whether is it likely the other agent will do all the work, etc). While our initial communicative action schemata are very simple, we believe they provide a lot of scope for extending them by more elaborate models of reasoning about communicative behaviour.

In the Colored Trails domain, agents play a simple game on a grid-like board with coloured squares such as the one shown in figure 2. Each player starts in a random position, and needs to move to a (randomly chosen) goal location. The rules of the game are simple: each player can only move to an adjacent square if she owns a chip of the same colour as that of the square, and the agent "spends" that chip when moving into the square. In this domain, we considered a setting of incomplete information; agents sometimes know that the other agent has a chip, but they do not always know its colour. Our problem instance is shown in figure 2 and involves two players, 5 chips, 4 colours, and a 4x4 board.

In Colored Trails, players have to exchange chips to obtain the chips they need to reach their goal square. Considering that we only want our agents to reach their goals, it would be easy for them to agree on a exchange if there was full observability. To add an extra layer of difficulty and to show the coverage of our model, we introduce partial observability by giving agent 0 information about agent 1's chips where one of the chips has an unknown color.

We start by looking at the example illustrated in figure 2. Agent 0 (me) is missing a light gray chip to reach its target, which Agent 1 (square) has. Agent 1 notices that it will not be able to fulfil the task by itself unless the other player transfers the unknown chip that might be light gray. Agent 0 generates a joint plan to determine what it can offer to agent 1 while still fulfilling its own goal, and discovers that agent 1 wants a white chip. The following dialogue is the result of this process of reasoning:

Agent 0: What is your chip 0 colour?
(ASK (CHIP CHI0 AGE1 UNKNOWN))

Agent 1: My chip 0 colour is light gray.
(REPLY (CHIP CHI0 AGE1 UNKNOWN) (CHIP CHI0 AGE1 GRA))

Agent 0: I will give you chip 2 in return for chip 0.
(PROPOSEIF TRANSFER AGE1 AGE0 GRA CHI0 TRANSFER AGE0 AGE1 GHO CHI2)

Agent 1: I accept this proposal.
(ACCEPT-PROPOSAL TRANSFER AGE1 AGE0 GRA CHI0)

As can be seen, agent 0 understood which chip she might need, asked about it, and then proposed to transfer a chip that it thinks agent 1 would want, in exchange for a chip that agent 0 needs. The communicative action schemata is the same as before, but our system is capable of filling in possible values for unknown parts of the situation based on the domain specification. Note that in situations with less knowledge, the agent might have to ask if its counterpart has any chip of colour gray, or any chip at all.

As a final example, we consider a situation where agents might have incorrect knowledge. Assume agent 1 (wrongly) thinks that agent 0 has a chip 5 of white colour. Agent 1 figures out that it needs a dark gray chip and a white chip to reach its goal. She proposes an exchange of her light gray chip for the other agent's dark gray chip. Agent 0 accepts this deal. Then, agent 1 proceeds by asking for the white chip 5. This chip, however, does not exist, thus agent 0 rejects this proposal. Agent 1 thereupon recalculates its plan, while considering that agent 0 will not give it the (nonexistent) chip 5. Agent 1 ends up generating another proposal for a white chip 2 that is accepted, which allows agent 1 to reach its goal. This is shown in the following dialogue obtained:

Agent 1: I will trade my light gray chip 0 for your dark gray chip 4.
(PROPOSEIF TRANSFER AGE0 AGE1 SEA CHI4 TRANSFER AGE1 AGE0 GRA CHI0)

Agent 0: I accept your proposal.
(ACCEPT-PROPOSAL TRANSFER AGE0 AGE1 SEA CHI4)

Agent 1: I propose that you transfer your white chip 5 to me.
(PROPOSE TRANSFER AGE0 AGE1 GHO CHI5)

Agent 0: I reject your proposal.
(REJECT-PROPOSAL TRANSFER AGE0 AGE1 GHO CHI5)

Agent 1: I propose that you transfer your white chip 2 to me.
(PROPOSE TRANSFER AGE0 AGE1 GHO CHI2)

Agent 0: I accept your proposal.
(ACCEPT-PROPOSAL TRANSFER AGE0 AGE1 GHO CHI2)

Despite the simplicity of these initial experiments, they already show that the same agent is able to deal with several different interaction situations. It is worth emphasizing that the planning-based dialogue generation code used in the above examples was not modified across the individual experiments, and that only the domain and task description given to the agents were adapted. This is a promising result in the sense that it may be possible to create agents that are more flexible in the long run, and employ more complex communication semantics encoded as planning theories. This is particularly important for flexible dialogue agent designs to be reusable across different specific applications.

5 Conclusions and Future Work

In this paper, we presented an initial design for dialogue agents that are able to automatically generate their own task-oriented utterances by using only domain and task descriptions, without a need for adapting the dialogue generation and understanding components involved.

Our design is based on a two-layered planning approach, which combines a domain-independent communicative action schemata available for the production of utterances, a similarly domain-independent algorithm for managing the overall dialogue process from an agent's point of view, and a standard planning component that is used to synthesise possible solutions for the domain-specific tasks of the agent. Initial experiments with this system showed that even with a fairly simple overall reasoning algorithm, our dialogue agents are capable of dealing with incomplete information and with different task domains the details of which are not known at the time of designing the communication language and protocol. Also, they can take assumptions about the other agent into account in their planning activity, and are able to employ replanning in case their original communicative plans fail.

Our research so far opens up interesting avenues for further work: One direction is to explore more complex models of planning, which would create conditional or contingent plans that would allow the agent to anticipate a whole range of opponent responses in a single dialogue planning step. This would enable the prediction of various expected reactions where there is uncertainty about the knowledge, intentions, and behaviours of the other agent in a single planning process, rather than predicting only one course of action and then replanning immediately. Another direction is to look at embedding more elaborate "theories of mind" in our communicative action schemata that would allow for a richer modelling of the participating agents' mental states to produce more reliable predictions, and at a wider set of communicative acts that would enable, for example, enquiry, argumentation, or negotiation dialogues among agents. The third and most ambitious direction would be to incorporate strategic reasoning methods in the dialogical framework. This would require decision- or game-theoretic extensions to the planning model, which are known to greatly increase the complexity of the underlying planning algorithms. An interesting path to explore with this regard would be the development of appropriate planning heuristics to be able to deal with this added complexity.

Acknowledgments. The research presented in this paper has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 567652 *ESSENCE: Evolution of Shared Semantics in Computational Environments* (`http://www.essence-network.com/`).

References

1. Chopra, A.K., Artikis, A., Bentahar, J., Colombetti, M., Dignum, F., Fornara, N., Jones, A.J.I., Singh, M.P., Yolum, P.: Research Directions in Agent Communication. ACM Transactions on Intelligent Systems and Technology. 4(2), 1–23 (2013)
2. Poslad, S.: Specifying Protocols for Multi-Agent Systems Interaction. ACM Transactions on Autonomous and Adaptive Systems. 2(4), 15–24 (2007)

3. Rosenfeld, A., Zuckerman, I., Segal-Halevi, E., Drein, O., Kraus, S.: NegoChat: a Chat-based Negotiation Agent. In: 13th International Conference on Autonomous Agents and Multiagent Systems, pp. 525–532. IFAAMAS Press, Paris (2014)
4. Rieser, V., Lemon, O.: Reinforcement Learning for Adaptive Dialogue Systems: a Data-Driven Methodology for Dialogue Management and Natural Language Generation. In: Theory and Applications of Natural Language Processing, Springer, Heidelberg (2011)
5. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann, San Francisco (2004)
6. Cohen, P.R., Perault, C.R..: Elements of a Plan-Based Theory of Speech Acts. Cognitive Science, 3, 177–212 (1979)
7. O'Brian, P.D., Nicol, R.C.: FIPA: Towards a Standard for Software Agents. BT Technology Journal. 16(3), 51–59 (1998)
8. Rich, C., Sidner, C.L.: COLLAGEN: A Collaboration Manager for Software Interface Agents. User Modeling and User-Adapted Interaction. 8(3-4), 315–350 (1998)
9. Moore, J., Paris, C.L.: Planning Text for Advisory Dialogues: Capturing Intentional and Rhetorical Information. Computational Linguistics. 19(4), 651–694 (1993)
10. Lambert, L., Carberry, S.: A Tripartite Plan-based Model of Dialogue. In: 29th Annual Meeting on Association for Computational Linguistics, pp. 47–54. ACL Press, Stroudsburg (1991)
11. Sklar, E. I., Azhar, M. Q., Flyr T., Parsons, S.: A Case for Argumentation to Enable Human-Robot Collaboration, In: Workshop on Argumentation in Multiagent Systems (ArgMAS) at Autonomous Agents and MultiAgent Systems (AAMAS), pp. 1–17, Springer-Verlag, Heiderberg (2013).
12. Giuliani, M., Petrick, R.P.A., Foster, M.E., Gashler, A., Isard, A., Pateraki, M., Sigalas, M.: Comparing Task-based and Socially Intelligent Behaviour in a Robot Bartender. In: 15th ACM on International Conference on Multimodal Interaction, pp. 263–270. ACM Press, New York (2013)
13. Nakano, M., Hasegawa, Y., Nakadai, K., Nakamura, T., Takeuchi, J., Torii, T., Tsujino, H., Kanda, N., Okuno, H.G.: A two-layer model for behavior and dialogue planning in conversational service robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3329–3335. IEEE Press, New York (2005)
14. Bylander, T.: The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence. 69(1), 165–204 (1994)
15. Hoffmann, J.: The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. Journal of Artificial Intelligence Research. 20, 291–341 (2003)
16. Hoffmann, J.: FF: The Fast-Forward Planning System. Artificial Intelligence. 22(3), 57 (2001)
17. Hoffmann, J., Nebel, B.: The FF Planning System: Fast Plan Generation through Heuristic Search. Journal Artificial Intelligence Research. 14(1), 253–302 (2001)
18. Blum, A.L., Furst, M.L.: Fast Planning through Planning Graph Analysis. Artificial Intelligence. 90(1), 281–300 (1997)
19. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D. PDDL-the Planning Domain Definition Language, `ftp://ftp.cs.yale.edu/pub/mcdermott/software/pddl.tar.gz` (2014)
20. Crosby, M., Petrick, R.P.A.: Centralised High-Level Planning for a Robot Fleet. In: Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2014), UK (2014)
21. Gal, Y., Grosz, B., Kraus, S., Pfeffer, A., Shieber, S. Colored Trails: a Formalism for Investigating Decision-Making in Strategic Environments. IJCAI 2005 Workshop on Reasoning, Representation, and Learning in Computer Games. pp. 25–30. AAAI Press, Menlo Park (2005)